European Commission

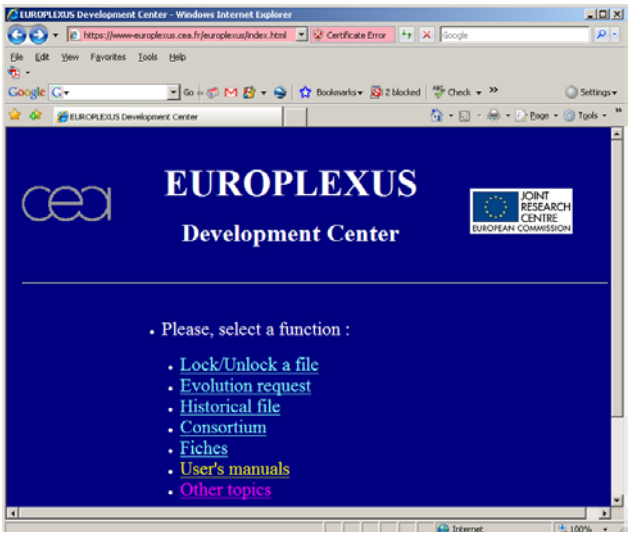
JRC TECHNICAL REPORTS

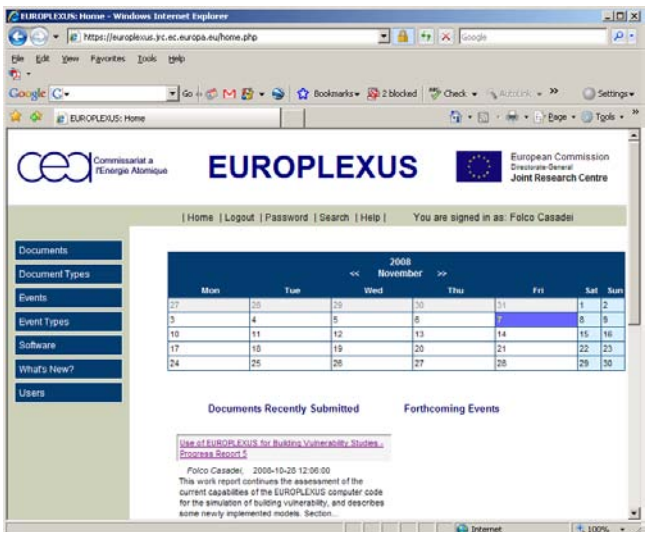
Organisation of the EUROPLEXUS Mirror Site (MS-Windows) at JRC Ispra Seventh Edition

F. Casadei
G. Solomos
G. Valsamos
H. Bung
A. Beccantini

2013

Report EUR 25821 EN





Joint Research Centre

European Commission
Joint Research Centre
Institute for the Protection and Security of the Citizen

Contact information
Folco Casadei
Address: Joint Research Centre, Via Enrico Fermi 2749, TP 480, 21027 Ispra (VA), Italy
E-mail: folco.casadei@jrc.ec.europa.eu
Tel.: +39 0332 78 9563
Fax: +39 0332 78 9049

http://ipsc.jrc.ec.europa.eu/
http://www.jrc.ec.europa.eu/

JRC79295

EUR 25821 EN

ISBN 978-92-79-28746-6

ISSN 1831-9424

doi:10.2788/8488

Luxembourg: Publications Office of the European Union, 2013

Printed in Italy

Contents

Preface to the Second Edition (January 2002 [11])

The scope of this document is to provide a comprehensive and detailed technical description of the EUROPLEXUS mirror site at JRC Ispra and, as such, it is subjected to frequent updates that reflect the continuous evolution of the software tools that are used for the managemenet and the development of the EUROPLEXUS code.

The First Edition [10], issued at the end of 2000, reflected the first version of the JRC mirror site but since then many improvements, additions, and changes have been applied, which prompted for the present, completely revised Second Edition.

The major changes are:

- The operating system at the JRC mirror site has changed from Windows NT 4.0 (various Service Pack levels) to Windows 2000 Professional (Service Pack 2).
- Compilers and (public domain) software tools have been updated to the most recent available versions.
- Many of the commands for code use and development (Section 5) and of the procedures for automatic code evolution (Section 6) have been endowed with new optional switches that further enhance their functionality and productivity.
- Several new commands and procedures have been added, including for example: `epx_filter_manual`, `epx_accents`, `epx_save`, `epx_test_man`, `epx_setat`, `epx_restore`.
- The User's manual sources (Section 0) are now subjected to filtering, exactly like the program sources themselves.
- The Section on interactive development (Section 8) has been enriched by a description of code profiling.
- A new Section (Section 10) on Implementation Notes has been added, which describes the gradual evolution of the mirror site implementation especially with respect to the evolution of the Fortran compiler.
- The default size of the stack for the executable has been set to 10 MB (0x640000) instead of 1 MB. Furthermore, a new switch `-s` has been added to the `epx_lk` command that allows to set the stack size when linking. The debug projects have also been modified accordingly.

References to previous versions (NT operating system) have been retained in the present edition whenever these have been thought to be useful for administrators of other mirror sites where that version of the Windows operating system is still in use.

Preface to the Third Edition (April 2003 [15])

The most important new development as concerns the site organization consists in the introduction of a new graphical module that performs advanced 3D rendering of the code results via OpenGL. A description is given in references [12, 13, 14]. This type of graphical output is meant to gradually replace the previous QuickWin-based graphics. The implications on the site organization are described in Section .10.10.

Other changes with respect to the previous Edition [11] are listed below.

- The `epx_bench` and `epx_test_benchmarks` procedures have been updated in order to solve the problem that the standard executable file could not be updated by the automatic evolution procedure if it was in use, see Section 10.6. A bug in the `epx_init` procedure has

been corrected, that prevented correct functioning when the local directory already contained a `main.ff` source file.

- The `epx_newer` (see Section **Error! Reference source not found.**), `epx_restore` (6.12), `epx_make` (6.13), `evototgz` (6.15) and `epx_ordo` (6.14) commands have been added. The latter procedure (`epx_ordo`) is now used internally in all scripts that need to find out the correct compilation order of Fortran 90 source files, in particular within `epx_cmp` and `epx_test_sources`.

- The `epx_evol_start` procedure has been updated to treat newly introduced optional `bm_*.zip` packages that may be associated to benchmark tests. Such zipped packages may contain any auxiliary files used for the benchmark test (e.g., CASTEM 2000 pre- or post-processing files `bm_*.dgibi`). The procedures `epx_get`, `epx_save` and `epx_restore` have also been updated to account for the new `.zip` file type.

- The whole JRC mirror site has been updated in January 2003, when the operating system has been changed from Windows 2000 Professional to Windows XP Professional. The installations and customizations are described in Section 10.7. New procedures `epx_ftp_getfiles`, `epx_ftp_putexe` and `epx_ftp_putman` have been added.

- The development of a Web site devoted to the EUROPLEXUS Consortium has been started. This is described in Section 10.9.

- The procedure `epx_vali.pl` (see Section 5.7) has been corrected to account for the case of abnormal run ends (in which the string `ARRET NORMAL` is missing on STDERR) and of multiple tests (and possibly multiple result qualifications) in a single run.

Preface to the Fourth Edition (September 2005 [16])

The most important changes are:

- A bug in procedure `epx_lk` has been corrected. The generation of the QuickWin executable was wrong because the `ifwin.obj` and `m_qwin.obj` files were not used at link time.

- A procedure `epx_deobso` has been added which performs the inverse operation of `epx_obsolete`, i.e. it restores a previously obsolete file.

- The BLAS (linear algebra system) package is placed in a separate library `Libblas.lib` that is included at link time. The `epx_lk`, `epx_save`, `epx_restore` and `epx_make` procedures have been modified accordingly.

- A new command procedure `epx_update_header` has been added which resets the header line of a source file (or multiple files) to the one of the current standard version of the file.

- The package SPLIB containing linear system solvers alternative to the standard Cholesky's method has been added, see Section 10.12.3.

- The filtering program `epx_filter` has been upgraded. The major modification is that now the filter accepts nested conditionals (at a single nesting level though), and this allows to considerably simplify some sources that depend both upon the platform type and on the type of graphical output available.

- In January 2005 the "atelier logiciel" has been updated by adding a new set of source files: validation tests. These are code validation examples typically having a stronger physical meaning than the benchmark (non-regression) tests, but also requiring longer CPU times. They are disseminated to all mirror sites and made available to users, but they are not

executed automatically as part of the evolution process. All relevant procedures have been updated to account for the new file types.

- The `epx_ordo` utility has been corrected.

- In July/August 2005 the mirror site has been completely re-installed due to a malfunctioning of the host machine which has made it necessary the re-installation of the operating system and of all the software. The opportunity has been taken to upgrade the various development tools, most notably the code development environment (now .NET 2003) and the Fortran compiler (now Intel 9.0). This has entailed numerous modifications to procedures, scripts etc. which are detailed in a dedicated Section (10.13) at the end of this report.

Preface to the Fifth Edition (October 2007 [17])

The most important changes are:

- A correction in the suggested setting for the EUROPLEXUS environment variable has been made (see Table 10 and description in Section 10.13.23).

- The *ad-hoc* Glut package version used in EUROPLEXUS has been modified so as to produce a static library (`Glut32.lib`) instead of a dynamic link library (`Glut32.dll`). This may facilitate the implementation of the code.

- A new command `epx_correct_ps` has been added which corrects the line thickness and font size of PostScript files issued from EUROPLEXUS. By using thicker lines and larger fonts, the legibility of graphs embedded in publications is ameliorated, especially when they are transformed into bitmaps.

- Two new procedures `stat_anom` and `stat_deve` have been added for the automatic filtering of the "Fiches d'anomalie" and "Fiches de developpement" files. Note that these require the (local) presence of the "Fiches" files, so they are actually used on the central site (atelier logiciel) and not at the mirror sites. They produce a succint resume of all open and closed fiche files, that will be shortly made available to developers by new buttons in the atelier logiciel.

- A new strategy has been implemented in the evolution driver procedure `epx_evol_start` (see Section 10.14.3). When an evolution involves a sources package and this contains at least one module file, all source files are re-compiled. This ensures better quality assurance of the code development process and largely reduces human effort when developing or modifying Fortran 90 module files.

- A new switch `-L` has been added to the `epx_lk` command. In this case the standard library (`Libplex.lib`) is ignored and completely re-built locally. The switch is therefore to be used only during evolutions involving a full recompilation of the code (see previous point).

- The command `epx_modrec` has been rendered much faster by avoiding redundant repeated generation of files. This has also improved a lot the efficiency of the `epx_modtree` command, which is used heavily in local development of Fortran 90 module files.

- A new command `epx_get_users` has been developed that allows to retrieve on the current directory all "users", both direct and indirect, of a given module file, i.e. all program units that use that module. This command has to be used in conjunction with the `epx_modtree` command.

- The command `epx_get_includers` has been corrected. An includer file may be already present on the evolution directory without causing an error.

- A new version of the Intel Fortran compiler (9.1) has been installed, see Sections 10.14.4 and 10.14.5.

- The procedure for the transfer of evolution packages `epx_evol_getfiles` has been modified, see Section 10.14.6, following the installation of the EUROPLEXUS Consortium web site on a new server, which has more restricted access options than before (no `ftp`). For the same reason, the `epx_ftp_putexe` and `epx_ftp_putman` procedures that are used to update the executable and the user's manual on the EUROPLEXUS Consortium web site have also been modified. Furthermore, an optional switch `-p` (for "patch") has been added to the `epx_evol_start` procedure, see Section 10.14.7.

- New scripts `epx_schtasks` and `epx_checkat` have been added, see Section 10.14.10. The former one replaces the `epx_setat` script. To solve a problem related to the functioning of the new versions of `epx_ftp_putexe` and `epx_ftp_putman` procedures, the evolution procedure must be launched as a specific user (folco) instead of as SYSTEM user, and this may not be done by the `at` command, but requires the more performing `schtasks` command.

- A new command `epx_dup_bench` was added, that duplicates an EUROPLEXUS benchmark (file `.dgibi` plus one or more files `.epx`) under a new name, see Section 10.14.8.

- A new version of the Intel Fortran compiler (10.0) has been installed, and a new debugging tool (Intel Vtune) has been made available see Sections 10.14.11 to 10.14.19.

- The EUROPLEXUS mirror site at JRC has been migrated to a new machine (sm47) for enhanced security.

- Option switches have been added to the `epx_cmp` and `epx_lk` commands that allow run-time checking of many common error sources such as use of uninitialized variables or illegal (out-of-bounds) array addressing.

Preface to the Sixth Edition (November 2008 [18])

The most important changes are:

- Some improvements (new sub-commands, help, etc.) have been introduced in the following procedures: `epx_grep`, `epx_newer`, `epx_init` and `epx_diff`.

- A new command `epx_manual` has been added which visualizes (a copy of) the user's manual in PDF format.

- A new version of the BLAS library has been taken, and the LAPACK library has been added. Furthermore, use is now optionally made of the LIB_VTK_IO package for binary output in VTK format for ParaView post-processing.

- Most importantly, the code has been ported to 64-bit operating system under MS-Windows (see Section 10.15). This includes the OpenGL graphical built-in post-processor and opens the way to much larger and memory-intensive simulations under Windows than was possible before with the 32-bit version.

Preface to the Seventh Edition (January 2013, present document)

- The command `exp_validate` has been updated so that it now executes all the input files contained in a certain validation package (`vl_*.zip`) and a new command `epx_validate_all` has been prepared, which allows to run a whole set of validation tests.

- The MPI parallel version of the code is now being automatically generated at each evolution of the code. To this end, the following procedures have been updated: `epx_cmp`, `epx_lk`, `epx_bench`, `epx_evol_start` and `epx_evol_64`.

- A "`-check`" version of the code executable is now constantly updated at every evolution on the 32-bit platform. To this end, the following procedures have been updated: `epx_cmp`, `epx_lk`, `epx_bench` and `epx_evol_start`. A new procedure `epx_evol_check` has been prepared which automatically executes the full benchmarks suite with the `-check` version every day. This version performs all runtime checks and is therefore considerably slower than the standard executable.

- The `epx_test_benchmarks` procedure has been updated: now a file containing all error messages of the performed benchmarks is produced, whose name is `epx_test_benchmarks.bad`. A new procedure `epx_make_evo` has been added, which facilitates the creation of the `evo.txt` file containing the list of the modified source files to be evolved. In addition, a file `lock.txt` is created, which can be used to lock all the files in the central mirror site. Finally, the epx_diff command has now a new switch `-t` which compares the listings (same as `-l`) but without displaying the different lines which contain the date and time.

- The `epx_cmp` procedure has been modified: the `-w` switch now activates all warnings insted of activation only warnings related to argument checking, declarations and truncated source.

- The `epx_make_evo` procedure has been corrected.

- The `epx_test_evo` procedure has been added. It checks an evolution package.

- The `epx_evol_start` procedure has been updated by adding three new optional switches (`-noqw`, `-nompi`, `-nocheck`) which allow to get faster evolutions (by skipping the QuickWin, MPI and –check versions) in case of emergency (e.g. blockage of several evolution packages).

- In June 2010 the Fortran compiler was upgraded to Version 11.1.060. The 32-bit machine was replaced by a new PC under the Windows 7 operating system. The EUROPLEXUS server (32-bit) was replaced by a more powerful machine. All executable versions built (4 versions under 32-bit and 2 versions under 64-bit) are automatically uploaded on the Consortium web site.

- In June 2011 use is made by default of the Intel MKL libraries instead of the locally generated Libblas and Liblapack libraries. The `epx_cmp` and `epx_lk` commands are modifiued accordingly. The debugging solutions of MS Visual Studio automatically copied by the `epx_init` command are also modified accordingly.

- In January 2013 the state of the mirror site at JRC has been summarized in Section 10.19, just befor retirement of F. Casadei. The Visual Studio (2010) and Intel Fortran compiler (3013) environments have been updated, as reported in Section 10.20.

1 Introduction

This document describes the organisation of a "mirror site" for the development of the EUROPLEXUS computer code at the Joint Research Center (JRC) of the European Commission (EC) at Ispra.

EUROPLEXUS is a computer code jointly developed by the French Commissariat à l'Energie Atomique (CEA DMT Saclay) and by EC-JRC IPSC in the framework of a collaboration contract (JRC Ref. N. 15384-1999-10 S0ED ISP FR). The code application domain is the numerical simulation of fast transient phenomena such as explosions, crashes and impacts in complex three-dimensional fluid-structure systems.

The initial version of EUROPLEXUS is derived from two ancestor codes: CEA's CASTEM-PLEXUS [1] and the former joint CEA/JRC code PLEXIS-3C [2]. One of the most distinctive characteristics of the new product with respect to its ancestors is that it is developed simultaneously by a number of development teams located at different geographical sites (e.g. Saclay and Ispra).

Simultaneously means that there exists at any time a unique version of the EUROPLEXUS code, available at each of the sites. Therefore, code versions divergence problems that plagued former joint software projects (most notably PLEXIS-3C) and required costly periodic updatings will be avoided.

The main principles of the EUROPLEXUS project and the general architecture of multi-site development are outlined in document [3], which stems partially from ideas expressed in [4] in the framework of the former CASTEM-PLEXUS project.

The present document and reference [5] are intended to give detailed technical information on the new project. While reference [5] provides some guidelines for updating the software architecture by means of new programming constructs offered by the FORTRAN 90 language which has been adopted for the project, the present report concentrates on the aspects related to code management and synchronised evolution across the different mirror sites. This document is organised as follows:

- Section 2 describes briefly the major tools that have been chosen to organise and maintain the JRC mirror site. Apart from the compilers, these are all available in the public domain for a variety of platforms, in particular the chosen scripting language perl.

- Section 3 presents some aspects of programming perl scripts under Windows and highlights the differences with respect to the Unix environment.

- Section 4 presents an overview of the site directory tree, that must be common to all mirror sites.

- Section 5 lists the commands (mostly perl scripts) that are available to all code developers for everyday development of the local test version(s) of the code.

- Section 6 presents the perl procedures which take care of the automatic evolution of the code, following the evolution "packages" which are continuously sent by the central mirror site. These procedures are of course not available to developers, but are maintained and activated by the mirror site administrator.

- Section 0 describes the implementation of the User's manual, which is also part of the common environment.

- Section 8 gives an overview of the interactive development environment, that is performed under MS Developer Studio, and also offers the possibility of interactive code debugging.

- Section 9 is a summary of all used tools, mostly available in the public domain.

- Section 10 contains the implementation notes for the various upgrades of the used tools, most notably for the various versions of the Forran compiler.

All the procedures and other source files mentioned or described in Sections 2 to 6 are listed in the Appendix at the end of the present document.

2 Chosen tools

This Section describes in some detail the basic tools that have been chosen to set up and maintain the JRC mirror site.

2.1 Operating system

The EUROPLEXUS mirror site that has been set up at JRC runs under the MS-Windows operating system. More precisely, the initial implementation described in reference [10] was under NT 4.0, while more recently (December 2001) this has been replaced by Windows 2000 Professional.

2.2 Languages and compilers

The FORTRAN compiler used at the moment is COMPAQ Visual Fortran Professional Edition 6.1, and the C compiler is Microsoft Visual C++ 6.0 Enterprise Edition. Both are commercial products.

Compilations may be performed from the command line (which under Windows corresponds to opening a "command prompt" window), by means of the following commands:

- `df` is the FORTRAN 90 compiler, which is also FORTRAN 77 compatible;
- `cl` is the C/C++ compiler;
- `link` is the linker, but linkage is preferably executed from `df` itself;
- `lib` is the object library manager.

An alternative way of working is via the Microsoft Developer Studio software, which allows to develop a software project through a graphical user interface (GUI) rather than from the command line. The product offers an interface to FORTRAN and C compilers, among others. Software development is organised as a "project", which may in turn contain a hierarchy of sub-projects.

The former method is more convenient for automatised development, i.e. for continuous code updating in order to reflect the changes distributed from the central mirror site. However, the command-line based development environment does not offer, to the authors' knowledge, any way of interactively debugging the code.

The GUI-based method, on the other hand, offers a rich debugging environment, and is very well suited for local developments. But because of its graphical/interactive nature it does not lend itself to automatic code updating.

Therefore, the chosen strategy is to adopt a combination of both methods: command-line commands (driven from perl procedures, see Section 2.3) are used for the automatic code updating, as described in Section 5, while Developer Studio projects are used for debugging and local development, see Section 8. Of course, a careful organisation of the whole project is necessary in order to make the two approaches live together without conflicts.

2.3 Scripting language

The chosen scripting language is perl (Practical Extraction and Reporting Language). This language, originally developed by Larry Wall, is now well established in various computer architectures, including Unix and Windows, for example.

Public domain implementations of the language are available. At JRC the version denominated Active Perl, from ActiveState (http://www.activestate.com/) has been adopted, which is freely obtainable from the network. The currently implemented Version is 5.6.1.

The main advantage of using perl is its portability, which is much larger, for example, than that of other scripting languages (e.g. C- or Korn shell are mainly available only in Unix environments). While perl was originally developed under Unix, and its port under Windows is not complete (because some of the operators are simply impossible to translate due to basic differences in the operating systems architectures), we have found that the available features are largely sufficient to develop the procedures of interest for the EUROPLEXUS project.

Nevertheless, there are some idiosyncrasies in the perl functionality under Windows which render the programming somewhat more difficult and lengthy than under Unix. These are discussed in detail below (see Section 3).

2.4 Shell commands

Despite the fact that many Unix commands (e.g. `wc`, `sort`, `tail` etc., to name just a few) may be emulated under Windows by means of perl scripts, it remains that such scripts must be written and debugged, and their efficiency is low due to the fact that perl is an interpreted language.

Therefore, we have looked for (freely available) ports of the most useful Unix command under Windows. Several such ports may be found on the network.

2.4.1 The NT Resource Kit

The NT Resource Kit that comes with NT 4.0 contains a few Unix command ports such as `cat`, `ls`, `cp`, `mv`, `wc`, and the `vi` editor, for example. However, their functionality is doubtful. Typically, one may expect to have trouble with long file names, with input and output redirections, with large files etc. Therefore the use of these Unix-like commands is not advisable (there are however some other useful commands in the Resource Kit, see below).

2.4.2 The Cygwin approach

An intentionally rather complete port of a Unix-like environment under Windows is underway, known as the Cygnus project (Cygwin): http://sourceware.cygnus.com/cygwin/. This of course includes most of the Unix commands, but has in the authors' opinion the following drawbacks:

- Since, by quoting the Cygnus site: “The Cygwin tools are ports of the popular GNU development tools and utilities for Windows 95, 98, and NT. They function by using the Cygwin library which provides a UNIX-like API on top of the Win32 API”, they require the whole Cygwin environment to be installed and activated on the Windows machine;
- Being a (more or less complete) Unix emulation, this software is relatively slow;
- The installation is complex and hardly justifiable for the present project.

2.4.3 A native port

Fortunately, there exist other, less demanding, ports of at least the most useful Unix tools under Windows. One of these may be found at http://www.weihenstephan.de/~syring/win32/UnxUtils.html. By quoting the author of the port (Stephan Weihen?): “Here are some ports of common GNU utilities to native Win32. In this context, native means the executables do only depend on the Microsoft C-runtime (msvcrt.dll) and not an emulation layer like that provided by Cygwin tools.”

The package contains ports of the (Gnu versions of the) most useful Unix commands, about 108 commands being available. While not all commands work properly under Windows (e.g. the `su` or `ln` commands obviously have some problems), most of them do and, what is very important, do not present any problems with either long file names or input/output redirection.

For these important properties, these commands have been adopted for the implementation of the local mirror site.

2.5 Editors

The standard editors available under Windows NT, namely NotePad and WordPad, are in the authors' opinion rather weak for program development (but of course editor choice is one of the most "personal" matters).

For example, one of the drawbacks of WordPad is that it may terminate a file without the last end-of-line character. In other words, if one edits a file and does not explicitly type a <CR> after the last line, then this is not inserted automatically when the file is saved.

This behaviour may cause problems with respect to file treatment by tools written in FORTRAN. In fact, a FORTRAN record is defined as a sequence of characters or data terminated by a new line (<CR>) character. When reading such ill-closed files, FORTRAN programs typically fail.

2.5.1 The vim editor

For Unix-oriented minds, various ports of the standard Unix editor `vi` are available under Windows. As mentioned above, one version is available under the NT Resource Kit. However, this version is very poorly implemented, not all the usual `vi` commands being available. Furthermore, cut-and-paste operations do not work properly.

A much better Windows implementation (even superior perhaps to some Unix implementations) is the `vim` (`vi` Improved) editor, freely available from http://www.vim.org/. This contains all the standard command, plus some improved ones, but most importantly it installs without problems under Windows and may be so configured (during the guided installation procedure) that the usual `xterm` functionality for cut and paste works in the Unix familiar way. Selecting (highlighting) text by dragging it with the left mouse button copies it to a buffer; then, by pointing to a different location (even in a different `vim` window!) and clicking the central mouse button the buffer text is inserted.

3 Perl programming under Windows

As anticipated in the previous Section, working with perl under Windows presents some peculiarities, which are described hereafter. These include, among others, the assimilation of perl scripts to system commands, the treatment of input and output redirection and the management of errors.

3.1 Assimilating perl scripts to system commands

In order to be able to execute a perl script, say `myscript.pl`, as a Windows system command, i.e. by simply typing, from a Command Prompt window:

```
myscript [arguments …]
```

instead of the more cumbersome:

```
perl myscript.pl [arguments …]
```

it is necessary to perform the following two actions.

3.1.1 File association

[Note: this first action action was necessary with older versions of perl, but is automatically performed at installation for the newer versions of Active Perl (such as e.g. version 5.6.1). Thefollowing description is left only as a hint for site administrators that use older perl versions.]

First, the File Association mechanism of Windows is used to associate the perl interpreter to perl files (characterized by the extension `.pl`). By quoting the ActivePerl on-line help, this is accomplished as follows:

1. Open the My Computer icon on the Desktop. The My Computer window should appear.
2. From the ***View*** menu in the ***My Computer*** window, choose ***Options***. The ***Options*** dialog box appears.
3. In the ***Options*** dialog box, select the ***File Types*** tab.
4. Click the ***New Type*** button. The ***Add New File Type*** dialog box appears.
5. In the ***Description of type*** box, type "Perl Script".
6. In the ***Associated extension*** box, type ".pl".
7. Leave the ***Content Type (MIME)*** box blank.
8. Click the ***New*** button beneath the ***Actions*** list. The ***New Action*** dialog box will appear.
9. In the ***Action*** box, type "Open" (it is important to use this name for the action!).
10. In the ***Application used to perform action*** box, type [full path to perl]`\perl.exe %1 %*`, where [full path to perl] is the full path to perl.exe on your machine. If perl is in your path, you can put just perl.exe, but for esoteric reasons it is better to put the full path. Also, if the path to your interpreter includes spaces (like `C:\Program Files\perl5`) put in the DOS path instead (`C:\progra~1\perl5`).
11. Click ***OK*** to close the ***New Action*** dialog box.
12. Click ***OK*** to close the ***Add New File Type*** dialog box.
13. Click ***OK*** to close the ***Options*** dialog box.

You can test your association by double-clicking on a perl script in the Explorer window. If `perl.exe` starts and executes the script, things are OK.

On Windows NT 4.0, an alternative way is to type the following from the command line:

```
ASSOC .pl=PerlScript

FTYPE PerlScript=[full path to perl]\perl.exe %1 %*
```

3.1.2 Setting the PATHEXT variable

The second task to be accomplished is to make sure that the command interpreter recognizes the name of the perl script (even without the extension "`.pl`") as a command. Using the "shebang" method typical of Unix, i.e. introducing a line such as `#!/usr/bin/perl` at the beginning of the script is useless, since Windows does not recognize the shebang.

The suggested method is then:

1. Open your ***Control Panel → System → Advanced → Environment Variables*** tab (Control Panel, System, Environment tab under NT).
2. Under ***User Variables for <current user>*** edit or add the `PATHEXT` variable so that it contains the following string: "`.COM;.EXE;.BAT;.CMD;.PL`" (without the double quotes!).
3. Click on ***Set***, then on ***Apply***.
4. The functionality is available for newly created console windows.

The effect of the above setting is that when the system encounters a presumable command name (`myscript`), it first tries to find, on the currently set `PATH`, a command named `myscript.com`,

then `myscript.exe`, then … and finally if neither of the above is found it localizes and executes `myscript.pl` (by processing it through `perl.exe`, thanks to the file association performed in the first step).

Note: the settings described above are valid for simple users' or developers' machines, but not for the EUROPLEXUS site administration machine (`sm48` at JRC). In fact, the site administration machine has to run automatic evolution procedures as unattended processes during the night, via the `at` command (see Section 6.3.1). Processes run under `at` belong to the SYSTEM user, and "see" the environment set for this user. In particular, the available environment variables are those for the system user (system variables). Therefore, in the case of the site administration machine the point 2 above reads: Under "System Variables" edit …. Furthermore, note that the machine must be rebooted for changes to system variables to enter into effect.

3.2 Input and output redirection

A major problem with the execution of perl scripts under NT is that input and (which is perhaps more important) output redirection do not work. In other words, assuming that one writes a perl script `hello.pl` that simply prints "Hello, world!". Then, executing it (after having set the environment as described in Section 3.1) on the command line as:

```
hello
```

produces, in the console window, the correct output:

```
Hello, world!
```

However, if one types:

```
hello >hello.txt
```

the file `hello.txt` is created, but is empty after execution of the command.

3.2.1 The pl2bat utility

This is of course a disastrous situation in the perspective of using perl for writing complex, automatic code evolution procedures. Fortunately, a solution to this problem exists and consists in transforming the perl script (file `.pl`) into a DOS "batch" file (`.bat`).

This task is accomplished in a totally automatic way by a utility, `pl2bat`, which is contained in the ActivePerl distribution. In order to produce file `myscript.bat` from file `myscript.pl`, simply type:

```
pl2bat myscript.pl
```

In practice, the utility "wraps" the perl script within a batch language header and trailer. This renders the script execution considerably slower than the original perl script, but solves the output redirection problem. The `pl2bat` utility admits various configuration options, but none were used for the present development.

Note that when both files `myscript.pl` and `myscript.bat` are present in the path, the latter is executed thanks to precedence in the `PATHEXT` variable setting (see Section 3.1.2). Therefore, it is safe to leave both files together.

Continuing the above example, then, if we type:

```
pl2bat hello.pl
hello >hello.txt
```

then we do obtain a non-empty output file.

3.2.2 Removing output bufferization

In the case of nested perl scripts, the transformation of these into batch files is not sufficient to ensure a proper redirection of output. In fact, since both the standard error (STDERR) and the standard output (STDOUT) are buffered by default in perl, the order in which output text from nested scripts comes out in the case of STDOUT or STDERR redirection to a file may look "messed up". Part of the ouput from a nested script may remain in the buffer until the main script terminates, and appears at the end of the file.

To avoid this effect it is possible to turn off STDOUT and STDERR bufferization. This further decreases the overall efficiency of scripts (which is already low due to interpretation), but is mandatory in case of automatised procedures. The only way of checking an automatic procedure result is in fact by examining its log file.

The perl code for obtaining this effect is as follows:

```
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
```

This code is inserted near the beginning of any perl script that may be used in nested scripts.

3.3 Testing for truth

In the authors' experience, one of the most confusing issues for beginner perl programmers is testing for truth within a perl script. This subject is, however, essential for controlling the flow of operations and especially for the management of errors, and deserves full attention.

The confusion comes mainly from the somewhat different treatment of various types of items in perl scripts: for example, direct tests within perl, or return codes from perl operators, from system commands, or from FORTRAN programs, or from external perl scripts. These are all shortly examined below.

3.3.1 Testing within perl

Direct testing of variables in a perl script obeys the following rules:

- The number `0` represents `false`;
- Any non-zero number represents `true`;
- A null (empty) string (`""`) is considered `false`;
- A non-null string is considered `true`; however, since a string containing only digits is converted to a number before interpretation, the string `"0"` is considered `false`.

This is illustrated by the following example:

```
$nullstr = "";
$str = "abc";
$zerostr = "0";
$zero = 0;
$one = 1;
#
if ($nullstr) {print "Null string is true\n"} else
              {print "Null string is false\n"};
if ($str)     {print "Non-null string is true\n"} else
              {print "Non-null string is false\n"};
```

```
if ($zerostr) {print "Zero (0) string is true\n"} else
              {print "Zero (0) string is false\n"};
if ($zero)    {print "Zero number is true\n"} else
              {print "Zero number is false\n"};
if ($one)     {print "One number is true\n"} else
              {print "One number is false\n"};
exit;
```

which produces:

```
Null string is false
Non-null string is true
Zero (0) string is false
Zero number is false
One number is true
```

3.3.2 Testing return codes from perl operators

Perl operators return `0` (`false`) if an error occurs, else they return a non-zero value (`true`). The following example illustrates this behaviour:

```
$gooddir = "good";      # Exists
$baddir = "bad";        # Does not exist
opendir (DIR, $gooddir) || die "Can't open $gooddir!\007\n";
print "Directory $gooddir opened.\n";
opendir (DIR, $baddir) || die "Can't open $baddir!\007\n";
print "Directory $baddir opened.\n";
exit;
```

which produces

```
Directory good opened.
Can't open bad!
```

3.3.3 Testing return codes from system commands

When the `system(cmd)` perl operator is used (or alternatively, the "backtick" form `` `cmd` ``), the return code from the command is obtained. Normally, system commands return `0` when successful, or a non-zero number when an error has occurred. Note that this is exactly the opposite of perl operators illustrated in the previous Section.

An example illustrated this behaviour:

```
$gooddir = "good";      # Exists
$baddir = "bad";        # Does not exist
(! system ("dir $gooddir")) || die "Can't dir $gooddir!\007\n";
print "Dir $gooddir successful.\n";
(! system ("dir $baddir")) || die "Can't dir $baddir!\007\n";
print "Dir $baddir successful.\n";
exit;
```

produces:

```
 Volume in drive E has no label.
 Volume Serial Number is 5C75-7910

Directory of E:\EUROPLEXUS\Fromcentral_old\prova\good

24/01/00  10:29         <DIR>          .
```

```
24/01/00  10:29         <DIR>          ..
               2 File(s)              0 bytes
                          3,219,804,160 bytes free
Dir good successful.


 Volume in drive E has no label.
 Volume Serial Number is 5C75-7910

 Directory of E:\EUROPLEXUS\Fromcentral_old\prova

File Not Found
Can't dir bad!
```

Note that, because of the "inverted" convention with respect to internal perl operators, the syntax is typically:

```
(! system ("cmd")) || die "Error message";
```

if the only thing to do in case of error is terminate the process (`die`) with an appropriate message, or alternatively:

```
if (system ("cmd")) {
  do_something …;
  die "Error message";
}
```

when dealing with the error requires several commands.

3.3.4 Testing return codes from FORTRAN programs

According to COMPAQ Fortran documentation (but this should be a standard for all Fortran compilers), the STOP instruction from a Fortran program has the following syntax:

```
STOP [message]
```

Where the optional `message`, if specified, must be either a character constant (Fortran terminology for a string) or an integer constant in the range `0` through `99999`.

If the message is omitted, the following default message is displayed:

```
STOP - Program terminated.
```

If `message` is a character constant, then:

- The message is displayed (actually it goes to `STDERR`, i.e. to logical unit number `0` by default; `STDIN` is unit `5` by default and `STDERR` is unit `6` by default);
- The program returns the code `0` to the operating system, for use by program units that retrieve status information.

If `message` is a number, then:

- The words `Return code`, followed by the number, are displayed.
- The program returns the least significant byte of that integer value (i.e., a value in the range `0` to `255`) to the operating system.

The run-time subroutine EXIT also terminates the program, flushes and closes all files, and returns control to the operating system.

The following examples illustrate this behaviour. Given the three Fortran programs:

```
program stop0
stop 0
end

program stop1
stop 1
end

program stopstr
stop 'ABC'
end
```

and the following perl script:

```
if (system ("stop0")) {print "Stop0 returned successful.\n"} else
                      {print "Stop0 returned an error!\007\n"}
if (system ("stop1")) {print "Stop1 returned successful.\n"} else
                      {print "Stop1 returned an error!\007\n"}
if (system ("stopstr")) {print "Stopstr returned successful.\n"} else
                      {print "Stopstr returned an error!\007\n"}
exit;
```

the result is:

```
0
Stop0 returned an error!
1
Stop1 returned successful.
ABC
Stopstr returned an error!
```

Note that, in analogy to the previous Section, the default return code (`0`) of the Fortran program is interpreted as an error by the perl script.

The behaviour of the program returning a string could be surprising, if one recalls that (non-null) strings are interpreted as `true` in perl. The fact is, as mentioned above, that the Fortran code uses the string only for display (it is sent to `STDERR`), but in fact returns `0` to the operating system, and this is interpreted as an error by perl.

In practice it is advisable to use only the two forms `STOP 0` or `STOP 1` in Fortran programs whose return code must be inspected by perl scripts. For example, in the `epx_filter` program (see Section 5.1) the choice made is to terminate by `STOP 0` in case of an error, and with `STOP 1` in case of success. This conforms to perl conventions (but is opposite to system command conventions and most likely to other Fortran programs).

3.3.5 Testing return codes from perl scripts

The final case of interest is that of an external perl script called from a "driver" perl script via the `system()` perl operator (or by the backtick syntax). This is useful for example in the case of complex perl procedures consisting of a nested hierarchy of perl scripts.

Unfortunately, tests (under Windows) indicate that the behaviour is somewhat unpredictable, depending on whether the called scripts terminated with the `exit()` perl funtion or with the `die`

operator. Furthermore, the return code from a script changes depending on whether the script is a perl script or a batch file obtained from a perl script via the `pl2bat` utility.

A detailed description of the problem, and a proposed solution, which however considerably complicates the scripts, is presented in Section 3.4.

3.4 Management of errors

Although the transformation from perl to batch cures output redirection, it unfortunately introduces a nasty side-effect: the command return code normally returned by any system command and also by perl scripts when executed as system commands is "lost" or corrupted when the script is transformed and executed in its batch form.

A simple example illustrates this behaviour. Given two simple perl scripts:

```
perllive.pl:  print "This is perllive\n";
perldie.pl:   print "This is perldie\n"; die;
```

and another script, which 'calls' these two via `system()`:

```
main.pl:
if (! system ("perllive")) {print "Perllive OK.\n"} else
{print "Perllive FAILED!\007\n"}
if (! system ("perldie"))  {print "Perldie OK.\n"} else
{print "Perldie FAILED!\007\n"} exit 0;
```

If one does not convert the scripts into `.bat` and types the command

```
main >out.txt 2>&1
```

then one gets (in file `out.txt`), the correct answers:

```
Perllive OK.
Perldie FAILED!
```

but the output from the two called scripts (i.e. the messages "`This is perllive`" and "`This is perldie`") is lost.

On the other hand, if one does convert the `perllive.pl` and `perldie.pl` scripts to `.bat`, and types the same command as above

```
main >out.txt 2>&1
```

then one gets (in file `out.txt`) the full output, both from the calling and from the called scripts, but with the wrong return code from the `perldie` script:

```
This is perllive
Perllive OK.
This is perldie
Died at perldie.bat line xxx.
Perldie OK.                        <---- Wrong!
```

This behaviour is very disturbing in setting up complex, nested perl scripts, because it does not allow to know, from a calling script, whether the called script has worked correctly or not. As a temporary way of circumventing the problem, in the EUROPLEXUS procedures use has been made of (permanent) error files.

Whenever a perl script that can be called from a higher-lever perl script and needs output redirection (and hence is transformed into batch) produces an error, it writes a corresponding error message to a (newly created) error file, that has the same base name as the script. The caller script may test the

correct termination of the lower-level one by checking the existence of the error file, rather than by inspecting the return code. This solves the problem but renders the scripts much more complex than necessary.

3.5 Sending e-mail from a perl script

The automatic code evolution procedure needs to send e-mail after each evolution:

- In case of success, the evolution log file is sent to both the central mirror site administrator and to the local site administrator;
- In case of failure, an error file is sent to both the central mirror site administrator and to the local site administrator.

E-mail can be sent from a perl script in a variety of ways. The method chosen here makes use of the `Mail::Sendmail` perl package, which is not part of the standard ActivePerl distribution but may be freely downloaded from the network, for example from the CPAN (Comprehensive Perl Archive on the Network: http://www.perl.com/CPAN-local/modules/index.html).

Installation of this package consists in editing the perl module file `sendmail.pm` and copying it to an appropriate directory in the perl installation (at the JRC site this is: `C:\Appl\Perl\lib\Mail`). The only necessary change in the file is setting the local mail server, which for the JRC site is `isis-ms.jrc.it`. After modification, the beginning of the module file will look as follows:

```
…
# *************** Configuration you may want to change ******************
# You probably want to set your SMTP server here (unless you specify it in
# every script), and leave the rest as is. See pod documentation for details

%mailcfg = (
    # List of SMTP servers:
    #'smtp'    => [ qw( localhost smtp.site1.csi.com ) ],
    'smtp'    => [ qw( isis-ms.jrc.it ) ],
    #'smtp'    => [ qw( mail.mydomain.com ) ], # example
…
```

For an example of use of the package, see the `epx_mail.pl` procedure listed in the Appendix. To test if the command works properly, do the following:

- On a local directory, prepare a file `coco.txt` containing the following text:

```
This is a test, please ignore it.
Folco
```

- Type the command: `epx_mail TEST 0 coco.txt` and verify that the mail has been sent.

3.6 Customizing the Date package

When installing perl on the machine, it is necessary to customize the package `date.pm` which is found in directory `Perl\site\lib\HTTP`. The modification consists in exporting the function `time2iso`, which is used in the automatic evolution procedure `epx_evol_start` (see Section 6.3). The line:

```
@EXPORT = qw(time2str str2time);
```

(line 8 in the current perl distribution, 5.6.0.618) should be modified as follows:

```
@EXPORT = qw(time2str str2time time2iso);
```

4 Site directory tree

The EUROPLEXUS mirror site at JRC consists of a "main" directory, containing a hierarchy of sub-directories. Some of these sub-directories are mandatory, others are optional.

Mandatory sub-directories have fixed names and must be present at each mirror site, since they contain the current EUROPLEXUS source files version, which is by definition unique and common on all sites.

Optional subdirectories contain accessory or temporary information. Their names and organisation are left to the preferences of local site administrators, but it would nevertheless be desirable to reach a certain level of standardisation.

4.1 Main directory and environment variables

The name of the main directory must be `EUROPLEXUS`, but its location on the local machine file system is arbitrary. To unify the development procedures, the main directory full path is not hard-coded in the procedures themselves, but is rather accessed via an environment variable which is also named `EUROPLEXUS`. The syntax for accessing the variable at the operating system level depends on the platform: for example under Unix it is typically `$EUROPLEXUS`, while under Windows it is `%EUROPLEXUS%`.

4.1.1 Setting the EUROPLEXUS environment variable

Under Windows, it is necessary that each developer (and also each simple user) of the code sets the value of the `EUROPLEXUS` variable on his machine, in order to get access to the necessary commands. This is done as follows:

1. Open your ***Control Panel → System → Environment*** tab.
2. Under ***User Variables for <current user>*** edit or add the `EUROPLEXUS` variable so that it contains the following string: "`\\SMNT04\E\EUROPLEXUS`" (without the double quotes!).
3. Click on ***Set***, then on ***Apply***.
4. The functionality is available for newly created console windows.

Note: the settings described above are valid for simple users' or developers' machines, but not for the EUROPLEXUS site administration machine (`sm48` at JRC). In fact, the site administration machine has to run automatic evolution procedures as unattended processes during the night, via the `at` command (see Section 6.3.1). Processes run under `at` belong to the SYSTEM user, and "see" the environment set for this user. In particular, the available environment variables are those for the system user (system variables). Therefore, in the case of the site administration machine the point 2 above reads: Under "System Variables" edit …. Furthermore, note that the machine must be rebooted for changes to system variables to enter into effect.

4.1.2 Setting the PATH

The EUROPLEXUS specific commands are located in sub-directory `util` of the main directory (see Section 4.2 below). In order to render these commands accessible, the `PATH` environment variable must be modified:

1. Open your ***Control Panel → System → Environment*** tab.
2. Under ***User Variables for <current user>*** edit or add the `PATH` variable so that it contains the following string: "…`;%EUROPLEXUS%\UTIL;`…" (without the double quotes!). The ellipses mean that there may be of course other directories on your path, before and/or after the EUROPLEXUS one.

3. Click on ***Set***, then on ***Apply***.

4. The functionality is available for newly created console windows.

4.2 Mandatory sub-directories

The main EUROPLEXUS directory must contain the mandatory sub-directories listed in Table 1. The contents of each sub-directory is briefly described in the Table.

Table 1 - The EUROPLEXUS mandatory sub-directories

Directory	Contents	Extension(s)
source	Code FORTRAN source files (subroutines, functions, modules)	`.ff`
source_c	Code C source files	`.c`
include	Code include files	`.inc`
bench	Benchmark input files (for non-regression tests)	`.epx`, `.msh`, `.zip`
manual	Manual files	`.ttx`
manual_filtered	Filtered manual files	`.tex`
util	Commands and procedures	`.pl`, `.bat`
validate	Validation examples	`.vld`, `.zip`

Note that the input files contained in the `bench` sub-directory are those to be used for non-regression tests during the automatic evolution of the code version. These are typically very short test cases, but which ideally cover all the code models so as to make sure that code development does not preclude the correct functioning of previously available models.

On the other hand, the input files contained in the `validate` sub-directory are more realistic code validation examples. These have a physical meaning, and typically require (much) longer CPU times than the non-regression tests. These files are collected and disseminated by the "atelier logiciel", so that the contents of this subdirectory is continuously synchronized among the various mirror sites. However, unlike the non-regression tests, they are not automatically executed at each evolution, since the process would be too long in terms of CPU. It is left to the users of the various sites to run some or all of these tests from time to time in order to further validate the code.

Note that, although the `util` sub-directory is mandatory, its contents may vary from site to site because the actual commands and procedures depend upon the local platform.

4.3 Optional sub-directories

The JRC mirror site contains the optional sub-directories listed in Table 2. These reflect the organizational choices that have been made. The role of each of these sub-directories will be explained in detail in Section 6, where the procedures for automatic code upgrading are described.

Table 2 - The EUROPLEXUS optional sub-directories (at JRC)

Directory	Contents	Extension(s)
backup	Backup of evolution packages received from the central site	`.tar.gz`, `.txt`
exe	Executable file(s)	`.exe`
fromcentral	Used for automatic evolution (normally empty)	
history	History files	`.his`
library	Libraries of object files (for the link)	`.lib`
module	Compiled F90 module files (`.mod`)	`.mod`

Directory	Contents	Extension(s)
tocentral	Material to be sent to central site (normally empty)	
trace	Traces of local evolutions	`.log`

5 Commands for code use and development

A number of commands that facilitate the everyday use and development of the EUROPLEXUS code at the local site have been developed. These are mostly perl scripts and are available to everybody at the local site, provided the settings described in Section 3.1 have been performed by the user.

Most of these commands are inspired by the commands that were used in the framework of the previous PLEXIS-3C project, see for example [6]. However, similar commands have been re-grouped into a single new command that accepts several optional switches. A list of commands is summarized in Table 3. Each command is then discussed in the following sub-sections.

Table 3 - Resume of commands for EUROPLEXUS development

Scope	Syntax	Section
Editing	`epx_vi [-i] [-h] [-b] [-m] [-u] [-l] [-p] [-v] file[.<ext>]`	5.2
Retrieving	`epx_get [-q] [-i] [-o] [-b] [-m] [-u] [-v] file[.<ext>]`	5.3
	`epx_get_users`	5.9.1
Compiling	`epx_cmp [-q] [-o] [-w] [-k key] [-x] [-g] [-c] . [-C] [-M] [-nomkl] [-times] file(s)[.ff]`	5.4
	`epx_ordo`	6.14
Linking	`epx_lk [-l|-L] [-o] [-w] [-c] [-f] [-p] [-e name] [-s size] [-M] [-nomkl]`	5.5
Running	`epx_bench [-e <executable>[.exe]] [-c] [-w] [-l] [-b] [-M nn] file[.epx]`	5.6
	`epx_validate [-e <executable>[.exe]] [-w] [-l] [-b] file[.vld]`	5.6.1
	`epx_validate_all [-o <owner>]`	5.6.1
	`epx_mkbatch`	6.5.1
Validating	`epx_vali file[.epx]`	5.7
Comparing	`epx_diff [-i] [-s] [-b] [-m] [-c] [-l] [-p] [-u] [-v] [-w] [-a] [-g] [-f] [-?] [-d dir] [-t] file(s)[.<ext>]`	5.8
	`epx_newer [-b] [-m] [-v] [-?]`	5.8
Searching	`epx_grep [-i] [-l] [-b] [-m] [-u] [-h] [-v] [-g] [-?] ["]pattern["]`	5.9
	`epx_modtree m_name`	5.9.1
Filtering	`epx_fil [-i<ext>] [-o<ext>] [-c[<char>]] [-p<pwd> ...] file(s)`	5.10
Filtering the manual	`epx_filter_manual [-q] [-k key] [-x] name(s)[.ttx]`	5.10.1
Filtering PostScript files	`epx_correct_ps [-t THICK] [-s SIZE] name(s)[.ps]`	5.10.2

Scope	Syntax	Section
Visualizing the user's manual	`epx_manual`	5.11

5.1 The epx_filter program

The `epx_filter` utility is a FORTRAN program that is used within the EUROPLEXUS project to accomplish the filtering of various (any) types of text (ASCII) files. Filtering is typically used to obtain a special version of the text file (for example, a FORTRAN subroutine file) that is adapted to a certain platform or operating system.

This is accomplished by inserting in the master version of the text file the text versions for all different choices, in the form of conditional constructs. The filter then, when supplied with appropriate password(s), produces a version of the text file that retains only the chosen parts.

An example may help clarify the use of the filter. Suppose that a certain subroutine `sub1.ff` contains a call to a system function and that the syntax of this call depends upon the operating system. Then, instead of having (and maintaining) separate versions of the subroutine `sub1`, one for each platform, say Unix, Windows, Cray, etc., one may simply write a master subroutine file `sub1.ff` such as:

```
      subroutine sub1(. . .)
      . . .
CIF UNIX
      <Text for Unix>
CELIF WIN
      <Text for Windows NT>
CELSE
      <Text for other platforms>
CENDIF
      . . .
      end subroutine sub1
```

In order to obtain the version `sub1.f` of the subroutine appropriate for, say, Windows, the filter would be invoked as follows:

```
epx_filter win <sub1.ff >sub1.f
```

Below is a complete description of the conditional syntax accepted by the filter and of its features. A listing of the filter program is provided in the Appendix.

5.1.1 Syntax of conditionals

The filter accepts three alternative syntaxes for the conditionals, that lead to slightly different results.

The first syntax corresponds to the above example and reads:

```
 CIF mot1 [mot2 …]
       [text1]
[CELIF mot3 [mot4 …]]
       [text2
[CELSE]
       [text3]
 CENDIF
```

Items in square brackets are optional. At most 8 “passwords” `mot1` etc. may be specified in the command line, each password being up to 16 characters in length (containing no spaces). Up to a

maximum of 8 or so CELIF branches may be specified, but of course only zero or one CELSE branches are accepted. The above syntax has the following effect:

- If the user specifies in the command line either password mot1 or mot2 or …, then text1 is activated (if any). Therefore, note that the specification of two or more passwords corresponds to a logical .OR. between them.
- Else, assuming that the CELIF clause is present and that the user specifies either mot3 or mot4 or …, then text2 is activated (if any);
- Else, assuming that the CELSE clause is present, text3 is activated (if any).

The present keywords are copied to the output file (since they begin with C they are interpreted as comments by FORTRAN compilers), but the inactive texts are replaced by the message C** ACCES INTERDIT ***. The compiler then "sees" only the active text.

The second syntax is a simple variation of the first one, where the first keyword is *IF instead of CIF:

```
 *IF mot1 [mot2 …]
       [text1]
[CELIF mot3 [mot4 …]]
       [text2
[CELSE]
       [text3]
 CENDIF
```

The only difference with respect to the first syntax is that "inactive" text branches are copied to the output file, by adding a comment character at position 1, instead of being replaced by the above mentioned message. This is useful in cases where all the information should remain visible in the filtered file, although properly commented out. Note that lines starting with * are also interpreted as comments by FORTRAN compilers.

The third syntax is of the form:

```
 CIFNOT mot1 [mot2 …]
       [text1]
[CELSE]
       [text2]
 CENDIF
```

It behaves like the first one, but is more handy in some cases. Again, the presence of two or more passwords corresponds to a logical .OR. between them.

It is very important to note that these conditionals may only be nested at nesting level 1, i.e., for example a CIF may contain another condition such as CIFNOT, but this may not contain a further one. Given the purpose of this facility, this is not considered as a serious limitation.

To improve legibility, the CELSE and CENDIF keywords may be optionally followe by comment (on the same line). However, in this case the comment must be separated from the keyword by at least one blank character.

5.1.2 Invoking the filter directly

The filter may be invoked directly from the command line as follows:

```
epx_filter [-c<char>] [mot1 [mot2 ...]] <input >output
```

where, for the first and third syntaxes of conditionals:

`-c<char>` This option, if present, causes the use of the given `<char>` in place of `C` in the output transcription of keywords, and of the built-in message `C** ACCES INTERDIT ***`, that is automatically output in place of the non-activated text blocks.

Example: '`-c%`' would give in output `%IF`, `%ELIF`, `%ELSE`, `%ENDIF` and `%** ACCES INTERDIT ***`. This example may be used for LaTeX documents, that use `%` as comment character.

`-c` If no character (i.e. a blank) is specified, then neither the keyword lines nor the built-in message are transcribed, so the output file will contain just the activated text.

For the second syntax of conditionals:

As already mentioned, the non-active branches, if any, are transcribed preceded by the chosen (or default) comment character. They remain therefore visible (but inactive) in the filtered text.

`-c<char>` This option, if present, causes the use of the given `<char>` in place of `C` in the output transcription of keywords, and of (commented) inactive branches.

`-c` No effect with this syntax.

The following rules apply to all syntaxes:

- Keywords `CIF`, `CELIF` etc. must start in column 1.
- Keywords are not case sensitive, i.e. `CIF`, `cif` or even `CiF` are the same.
- Passwords `mot1`, `mot2` etc. are not case sensitive, either, and may not contain blanks.

To avoid ambiguous cases, the filter performs the following tests while filtering the text file:

- Tests on the command line syntax:

  The user may not specify twice the same password (or equivalent passwords that differ only by letter case) on the command line.

- Tests on the input text file contents:

  Passwords in each branch of a conditional must differ from any passwords in other branches of the same conditional.

  If more than one password is specified, all passwords must belong to the same branch of a conditional.

  The `CIFNOT` syntax does not admit `CELIF` branches.

Finally, note that the program returns 1 if everything was OK, else it returns 0. This choice corresponds to the default for perl scripts (but is opposite to the behaviour of system commands!).

5.1.3 Installing the filter

Installing the filter requires some adaptation to the local platform. This has to be done manually, because automatic filtering is likely to be unavailable while installing the filter itself.

Compilation of the filter source requires a FORTRAN 90 compiler. This should not be a problem because the same compiler is also needed for EUROPLEXUS. The filter needs accessing the arguments (passwords) typed by the user on the command line when the filter is invoked.

For this purpose two system-related functions are usually available on any platform, although the syntax of these functions varies. The first function returns the number of arguments on the command line (e.g., `NARGS()` under Windows, `IARGC()` under AIX), the second one the arguments themselves (e.g. `CALL GETARG()` under both Windows and AIX, but with different exchange lists!).

The current source of the `epx_filter` program contains code appropriate for both the Windows and the IBM AIX platform. By default, the Windows code is active and the AIX code is commented out. To implement the filter under AIX, it is sufficient to comment out the Windows code and uncomment the AIX one. The code (just a few lines) position is marked by comments `C ATTENTION!` near the beginning of the program.

To port the filter to a different platform, please consult your FORTRAN manuals to find out the exact name and syntax of the above mentioned system functions.

5.2 Editing

The `epx_vi` command may be used to visualize a file from the EUROPLEXUS directory. Any ASCII file, including e.g. source (Fortran) files or include, or history files may be visualized. The chosen file is edited in read-only mode by the `vim` file editor (an improved version of Unix's `vi` ported to Windows, see Section 2.5) within a new pop-up text window.

The syntax reads:

```
epx_vi [-i] [-h] [-b] [-m] [-u] [-l] [-p] [-v] file[.<ext>]
```

where:

`-i` Option to visualize an include file (extension `.inc`), rather than a source file (extension `.ff`). By default, a source file is visualized.

`-h` Option to visualize a history file (extension `.his`), rather than a source file (extension `.ff`). By default, a source file is visualized.

`-b` Option to visualize a benchmark file (extension `.epx`), rather than a source file (extension `.ff`). By default, a source file is visualized.

`-m` Option to visualize a manual file (extension `.ttx`), rather than a source file (extension `.ff`). By default, a source file is visualized.

`-u` Option to visualize a utility file (extension `.pl`), rather than a source file (extension `.ff`). By default, a source file is visualized.

`-l` Option to visualize a listing file (extension `.listing`), rather than a source file (extension `.ff`). By default, a source file is visualized.

`-p` Option to visualize a PostScript file (extension `.ps`), rather than a source file (extension `.ff`). By default, a source file is visualized.

`-v` Option to visualize a validation file (extension `.vld`), rather than a source file (extension `.ff`). By default, a source file is visualized.

`file` Name of the file to be visualized. No extension is required. If the extension is given, it must match the setting of the previous option(s), if any.

Examples:

```
epx_vi celem
```

Opens a new pop-up `vim` window and edits (in read-only mode) the file `celem.ff` from the EUROPLEXUS library: `%EUROPLEXUS%\source\celem.ff`.

```
epx_vi -i contro
```

Opens a new pop-up `vim` window and edits (in read-only mode) the file `CONTRO.INC` from the EUROPLEXUS library: `%EUROPLEXUS%\include\CONTRO.INC`. Note that file names are case-insensitive under Windows.

```
epx_vi -h celem
```

Opens a new pop-up `vim` window and edits (in read-only mode) the file `celem.his` from the EUROPLEXUS library: `%EUROPLEXUS%\history\celem.his`. Note that file names are case-insensitive under Windows.

```
epx_vi sgdi.ff
```

Opens a new pop-up `vim` window and edits (in read-only mode) the file `sgdi.ff` from the EUROPLEXUS library: `%EUROPLEXUS%\source\sgdi.ff`.

5.3 Retrieving

The `epx_get` command may be used to retrieve a file from the EUROPLEXUS directory. Any relevant type of file may be retrieved, including: source (Fortran) files, include files, module (object) files, benchmark files, manual files, or utility files. The chosen file is copied to the current directory, but only if a file with the same name is not already present, else an error message is issued.

The syntax reads:

```
epx_get [-q] [-i] [-o] [-b] [-m] [-u] [-v] file[.<ext>]
```

where:

`-q` Quiet: do not echo (following) command line switches and do not confirm that file has been copied to current directory. Useful when the command is invoked from within a procedure.

`-i` Option to retrieve an include file (extension `.inc`), rather than a source file (extension `.ff`). By default, a source file is retrieved.

`-o` Option to retrieve an (object) module file (extension `.mod`), rather than a source file (extension `.ff`). By default, a source file is retrieved.

`-b` Option to retrieve a benchmark input file (extension `.epx`), rather than a source file (extension `.ff`). By default, a source file is retrieved. Note that if a corresponding benchmark mesh file (extension `.msh`) exists, then it is also retrieved. The same for a corresponding optional zip file (extension `.zip`).

`-m` Option to retrieve a manual file (extension `.ttx`), rather than a source file (extension `.ff`). By default, a source file is retrieved.

`-u` Option to retrieve a utility file (extension `.pl`), rather than a source file (extension `.ff`). By default, a source file is retrieved.

`-b` Option to retrieve a validation file (extension `.vld`), rather than a source file (extension `.ff`). By default, a source file is retrieved. Note that the corresponding mandatory zip file (extension `.zip`) is retrieved as well.

`file` Name of the file to be retrieved. No extension is required. If the extension is given, it must be either `.ff` or `.inc`, or `.mod`, or `.epx.` or `.ttx`, or `.pl` and it must match the setting of the preceding options. The file is retrieved only if a file with the same name does not exist in the current directory.

Examples:

```
epx_get celem
```

Copies the file `celem.ff` from the EUROPLEXUS library: `%EUROPLEXUS%\source\celem.ff` to the current directory (if `celem.ff` is not already present in the directory).

```
epx_get -i contro
```

Copies the file `CONTRO.INC` from the EUROPLEXUS library: `%EUROPLEXUS%\include\CONTRO.INC` to the current directory (if `CONTRO.INC` is not already present in the directory). Note that file names are case-insensitive under Windows.

```
epx_get -o m_fragment
```

Copies the file `m_fragment.mod` from the EUROPLEXUS library: `%EUROPLEXUS%\module\m_fragment.mod` to the current directory (if `m_fragment.mod` is not already present in the directory). Note that file names are case-insensitive under Windows.

```
epx_get -b bm_cir_bifur
```

Copies the file `bm_cir_bifur.epx` from the EUROPLEXUS library: `%EUROPLEXUS%\bench\bm_cir_bifur.epx` to the current directory (if `bm_cir_bifur.epx` is not already present in the directory). Since the mesh file `bm_cir_bifur.msh` is also present in the library, it is also copied to the current directory.

```
epx_get -m gbint_0050
```

Copies the file `gbint_0050.ttx` from the EUROPLEXUS library: `%EUROPLEXUS%\manual\gbint_0050.ttx` to the current directory (if `gbint_0050.ttx` is not already present in the directory).

5.4 Compiling

The `epx_cmp` command may be used to compile a Fortran file on the current directory by using includes and object module files (`.mod`) from the EUROPLEXUS directory. The switches `/automatic` (local variables are placed on the run-time stack) and `/traceback` (a symbolic rather than hexadecimal traceback is activated) are always used.

Note that compilation is automatically preceded by filtering (see Sections 5.10, 5.1). The filtering process transforms the `.ff` file into `.f`, and this file is then subjected to compilation. The filtering passwords appropriate for the local site are automatically passed to the filtering program. Currently at JRC the two passwords "`WIN OGL`" ("`WIN QWIN`" prior to May 2003) are used (hard-coded within the `epx_cmp` script), but these might change as the EUROPLEXUS version continues to evolve.

The syntax reads:

```
epx_cmp [-q] [-o] [-w] [-c] [-C] [-k key] [-x] [-g] [-M]
[-nomkl] [-times] [file(s)[.ff]]
```

where:

`-q` Quiet: do not echo (following) command line switches. Useful when the command is invoked from within a procedure.

`-o` Option to compile with optimization, i.e. by using the switches `optimize:5 /nodebug`. By default, the file is compiled in debug mode, i.e. by using `/nooptimize /debug:full`.

`-c` Option to compile with full run-time checks, i.e. by using the switch `/check:all`. It seems that it may be used also in conjunction with the `-o` switch. Attention: if used in conjunction with `-o` switch it must be specified **after** it, i.e. `epx_cmp -o -c` and **not** `epx_cmp -c -o`!

`-C` Same as `-c` but, in addition, if the file name to be compiled is `main`, then the procedure automatically adds also the `/fpe:0` and `/fp:strict` switches. However, this seems to create a lot of errors in the OpenGL library.

`-w` Option to compile with full warnings, i.e. by using the switches `/nooptimize /debug:full /warn:all`. These settings transform warnings in errors and are suited especially for testing routines that have to be evolved by the central mirror site. In particular, any undeclared variables produce an error (this is the same as imposing the presence of an `IMPLICIT NONE` directive in the source). By default, the file is compiled in debug mode, i.e. by using `/nooptimize /debug:full`.

`-k key` Add filtering key `key` to default filtering keys. This switch may be repeated to add as many keys as needed.

`-x` Suppress all default filtering keys (`WIN OGL`). This is sometimes useful in combination with the `-k` switch to get full control on the set of keys to be used during the filtering process.

`-g` Compile using QuickWin libraries instead of Static libraries (`/libs:qwin`). This option is required to compile any sources on the current directory when producing a QuickWin executable (new with Intel 9.0 Fortran compiler), and is internally (and automatically) invoked by the `epx_lk -w` command.

`-M` Compile for MPI: the extra keyword `MPI` is added automatically to the filtering system (same as specifying `-k MPI`), and the MPI include directory is added to the include search path.

`-nomkl` Do not use the Intel MKL libraries, use the local Libblas and Liblapack libraries instead.

`-times` Print starting and ending compilation times in the `.err` file. These can then be further processed by the `compil_times` command.

`file(s)` Name of the file (or files) to be compiled. No extension is required. If the extension is given, it must be `.ff`. Note that wildcards may be used to specify multiple file names. If omitted, then all source files (`*.ff`) present in the current directory are compiled. The compilation order is determined by running the `epx_ordo` command (which destroys any previous occurrence of `epx_ordo.txt` file), decribed in Section 6.14. At the end of compilation, the file `epx_ordo.txt` may be inspected to check the compilation order.

A log of compilation messages is produced, in file `file.err`. If compilation errors occur, a message prompts the user.

Examples:

```
epx_cmp
```

Compiles all files with an extension `.ff` in the current directory, in debug mode, and in the correct order (as produced by the `epx_ordo` utility).

```
epx_cmp -o
```

Compiles all files with an extension `.ff` in the current directory, with optimization, and in the correct order (as produced by the `epx_ordo` utility).

```
epx_cmp celem
```

Compiles the file `celem.ff` in the current directory, in debug mode.

```
epx_cmp -o celem
```

Compiles the file `celem.ff` in the current directory, with optimization.

```
epx_cmp -o *.ff
```

Compiles all files with an extension `.ff` in the current directory, with optimization.

```
epx_cmp -w *.ff
```

Compiles all files with an extension `.ff` in the current directory, with enhanced warnings (suitable for routines testing before evolution).

```
epx_cmp -w -k JRC *.ff
```

Compiles all files with an extension `.ff` in the current directory, with enhanced warnings and by using the additional key `JRC` for filtering. The set of filtering keys becomes therefore `WIN OGL JRC`.

```
epx_cmp -w -x -k JRC *.ff
```

Compiles all files with an extension `.ff` in the current directory, with enhanced warnings, by suppressing all default filtering keys and by using the additional key `JRC`. The set of filtering keys becomes therefore `JRC`.

5.5 Linking

The `epx_lk` command may be used to produce a local executable version of EUROPLEXUS. The link is done by invoking the FORTRAN compiler `df` with appropriate parameters for the linker.

The command links all object (`.obj`) files present in the current directory against the standard EUROPLEXUS libraries (`%EUROPLEXUS%\library\libplex.lib` for FORTRAN files, and `%EUROPLEXUS%\library\libplex_c.lib` for C files), and produces a local executable `epx.exe`.

Linking requires that the main object file (`main.obj`) be present in the current directory. If this is not the case, the procedure extracts `main.obj` from the FORTRAN library, then performs the link and finally removes `main.obj`.

The syntax is:

```
epx_lk [-l|-L] [-o] [-w] [-c] [-f] [-p] [-e name] [-s size] [-nomkl]
```

where:

`-l` Option to generate and use local library instead of the standard EUROPLEXUS library for FORTRAN files. The library `%EUROPLEXUS%\library\libplex.lib` is copied locally and then it is updated by all object files (`*.obj`) present in the local directory. This library is used to perform the link. This option is useful in two cases: 1) if the number of object files to be linked is very large, listing them all in one link command line may cause an error, which is avoided by using the modified library instead, and 2) if the link is to be performed during code evolution, the modified library remains at disposal (the name is `Libplex.lib`) and may directly replace the standard library after successful testing.

`-L` Same as `-l` but the standard library is ignored and a complete library is built up locally. This assumes that all sources have been re-compiled locally (full compilation). The switch is useful during code evolution when a complete re-compilation of the code is performed (see also the modifications in the evolution driver procedure `epx_evol_start`, Section 10.14.3). The two switches `-l` and `-L` are mutually exclusive.

`-o` Option to invoke `df` with optimization, i.e. by using the switches `optimize:5 /nodebug`. By default, `df` is invoked in debug mode, i.e. by using `/nooptimize /debug:full`.

`-w` Generate a QuickWin (Graphic) application instead of a Console (Text only) application.

`-c` Option to link with library generated by using the switch `/check:all`, which allows checking the entire code. This library is called `libplex_check.lib`.

`-f` Force creation of an executable even in case of link errors. This option may be useful in special cases, such as e.g. the creation of an ad-hoc "batch" (non-QuickWin) version of the code to execute the benchmarks suite.

`-e name` Use `name` as the name of the resulting executable file instead of the default value `epx.exe`.

`-p` Prepare the load module `epx.exe` for profiling (see Section 8.4). Produces an auxiliary file `epx.map`.

`-s size` Use `size` as the stack size instead of the default (10 MB). The size must be expressed in hexadecimal noation: for example, a 100 MB stack is obtained by the switch `-s 0x6400000`.

`-nomkl` Do not use the Intel MKL libraries, use the local Libblas and Liblapack libraries instead.

Examples:

```
epx_lk -o
```

Links all object files presnt in the current directory (if any) with the EUROPLEXUS libraries and produces the local executable `plex.exe`. The compiler `df` is invoked in optimized mode.

```
epx_lk -l -o
```

Copies the standard EUROPLEXUS library of FORTRAN files to the local directory and updates it with all locally present object files. Then it performs the link and produces the local executable `plex.exe`. The compiler `df` is invoked in optimized mode. The local modified library is left on the current directory under the name `Libplex.lib`.

```
epx_lk -w -o
```

Like the first example above, but produces a QuickWin application instead of a Console application. An alternative linking command, `epx_mkbatch`, that produces a special code version suitable for batch execution is described in Section 6.5.1.

5.6 Running

The `epx_bench` command may be used to run and validate the EUROPLEXUS code. The test input file (extension `.epx`) may be a local file or a file from the standard benchmarks directory `%EUROPLEXUS%\bench` (see Section 4.2). In the latter case, the input file and the associated mesh file (`.msh`), if any, are copied from the standard to the local directory.

The executable load module used to run the test is by default the current standard code executable: `%EUROPLEXUS%\exe\europlexus.exe`. A different load module (for example, a local development or test version) may be specified by a switch.

At the end of the run, the test case result is validated by automatically invoking the `epx_vali` command described in Section 5.7.

By default, the code is run in "interactive" mode: the STDOUT (usually containing the echo of input directives and some messages from the code) is not redirected to a file, and code interactive piloting becomes possible, provided the `CONV` directive is contained in the input file. This also gives access to interactive on-screen (and on-file) graphics.

Alternatively, the code may be run in "batch" mode by supplying the `[-b]` switch. In this case the STDOUT is redirected to a file `<base>.eco`. The `CONV` directive should not be present in the input file in this case, else the code will halt forever waiting for keyboard input. This execution mode is useful for the execution of benchmark test series, or for very long runs.

Note that in any case (both interactive and batch mode) the STDERR is redirected to a file `<base>.std`. This file is then analysed by the validation procedure, see Section 5.7.

For implementation details concerning the use of the current standard executable, see Section 10.6.

The syntax is:

```
epx_bench [-e <executable>[.exe]] [-c] [-w] [-l] [-b] [-M nn]
file[.epx]
```

where:

`-e`	Use the `<executable>` load module specified next to run the test (with or without the extension `.exe`), instead of the current standard code executable.
`-c`	Use the standard `-check` executable (named `europlexus_check.exe`), instead of the current standard code executable.
`-w`	Use the current standard QuickWin code executable instead of the current standard Console code executable.
`-l`	Use an input `file` from the current local directory rather than from the standard EUROPLEXUS benchmarks directory.

`-b` Run the code in "batch" mode (redirect STDOUT to a file with the extension `.eco`).

`-M nn` Run the MPI version of the code using `nn` processors (`nn` must be > 0). The input file must contain the definition of `nn` sub-domains. The MPI package (in particular the `mpiexec.exe` executable file) must be installed in the machine, see also Section 10.15.3: the file must reside under the default installation location (`C:\Program Files\MPICH2\bin`).

`file` Name of the input file (with or without the extension `.epx`).

Examples:

```
epx_bench bm_cir_bifur.epx
```

Run and validate, by using the current standard code executable, the benchmark test `bm_cir_bifur.epx` from the standard EUROPLEXUS benchmarks directory. The input file (and the associated mesh file `bm_cir_bifur.msh`) are copied to the current directory and executed locally.

```
epx_bench -l my_test
```

Run and validate, by using the current standard code executable, the test `my_test.epx` from the current directory.

```
epx_bench -e epx bm_cir_bifur
```

Run and validate, by using the local `epx.exe` executable, the benchmark test `bm_cir_bifur.epx` from the standard EUROPLEXUS benchmarks directory. The input file (and the associated mesh file `bm_cir_bifur.msh`) are copied to the current directory and executed locally.

```
epx_bench -b bm_cir_bifur.epx
```

Run in batch mode and validate, by using the current standard code executable, the benchmark test `bm_cir_bifur.epx` from the standard EUROPLEXUS benchmarks directory.

5.6.1 Running a validation test

A specific command, `epx_validate`, is available for running a so-called EUROPLEXUS validation test. These tests are normally contained in the `validate` sub-directory, and consist of a description file (with the extension `.vld`) and a zipped file (extension `.zip`) containing the input data, post-treatment data, associated documentation, etc.

The syntax is similar to that of the `epx_bench` command:

```
epx_validate [-e <executable>[.exe]] [-w] [-l] [-b] file[.vld]
```

where:

`-e` Use the `<executable>` load module specified next to run the test (with or without the extension `.exe`), instead of the current standard code executable.

`-w` Use the current standard QuickWin code executable instead of the current standard Console code executable.

`-l` Use an input `file` from the current local directory rather than from the standard EUROPLEXUS validation directory.

`-b` Run the code in "batch" mode (redirect STDOUT to a file with the extension `.eco`).

`file` Name of the input file (with or without the extension `.vld`).

Examples:

```
epx_validate vl_cea_jwls_2d.vld
```

Run and validate, by using the current standard code executable, the validation test `vl_cea_jwls_2d` from the standard EUROPLEXUS validation directory. The description file `vl_cea_jwls_2d.vld` and the associated zip file `vl_cea_jwls_2d.zip` are copied to the current directory, and the latter one is unpacked locally. The input file `vl_cea_jwls_2d.epx` (and the associated mesh file `vl_cea_jwls_2d.msh`) as extracted from the `.zip` file are used to run the test locally.

5.6.2 Running a whole series of validation tests

A specific command, `epx_validate_all`, is available for running a whle set of EUROPLEXUS validation test. These tests are normally contained in the `validate` sub-directory, and consist of a description file (with the extension `.vld`) and a zipped file (extension `.zip`) containing the input data, post-treatment data, associated documentation, etc.

The validation tests are run each in a separate sub-directory of the current directory, which is created ad-hoc and must not be pre-existing. The name of each sub-directory is the name of the validation test (without the `.vld` extension).

The syntax is:

```
epx_validate_all [-o <owner>]
```

where:

`-o` Use the specified `<owner>` name, instead of any name (*) to build up the validation test case names.

Examples:

```
epx_validate_all
```

Run and validate all validation tests present in the EUROPLEXUS validation directory. These are of the form `vl_*.vld`.

```
epx_validate_all -o jrc
```

Run and validate the validation tests present in the EUROPLEXUS validation directory which belong to the `jrc` "owner". These are of the form `vl_jrc*.vld`.

5.7 Validating

The `epx_vali` command may be used to verify the results of an EUROPLEXUS test that has been previously run. Two types of validation are currently possible:

- Validation based on run termination: the run is considered correct by the sole fact that the code terminates without an error message and the string `ARRET NORMAL` is printed to STDERR. No check whatsoever on the test results is performed. This is of course by no means a real validation and should be replaced whenever possible by the more precise validation method described below.

- Validation based on punctual test results at the final time, via the `VALI` or `QUAL` user input directive. This directive verifies one or more test results at the final calculation time, e.g. velocities

or other nodal quantities, stresses or other element quantities, at a series of points specified by the user. If the expected results, also specified in the input file, are found within a given tolerance, the test is considered valid and the string `VALIDATION: CORRECT` followed by `ARRET NORMAL` (on the following line) is printed to STDERR, else it fails and the string `VALIDATION: ERREUR` followed by `ARRET NORMAL` (on the following line) is printed to STDERR.

In addition, one should consider also the case that the code, for some reason, does not complete the calculation. In such cases the string `ARRET NORMAL` is not printed to STDERR. This is always considered as a non-valid run, even in the (unlikely) case that the error has occurred after validation and the validation was correct.

Finally, it should also be considered that there is the possibility to specify several test cases, separated by the SUIT directive, in the same input file. Each of them may have a final validation directive, which will produce either `VALIDATION: CORRECT` or `VALIDATION: ERREUR`. In any case, the final message `ARRET NORMAL` is printed just once, at the very end of the run.

The second method (with results qualification) is the preferred one. In particular, all non-regression tests should be gradually modified to use this validation method.

The `epx_vali` command simply inspects the STDERR file produced by the test run, detects any of the above specified messages, and returns an appropriate validation or error message. Its use as a stand-alone command should be rare, since it is automatically invoked by the `epx_bench` command, for example.

The syntax reads:

```
epx_vali file[.epx]
```

where:

`file` Name of the test file to be validated, with or without the extension `.epx`. The test must have been previously run and its standard error file (`file.std`) must be available for inspection.

Examples:

```
epx_vali my_test
```

Inspects the STDERR file `my_test.std` relative to test `my_test.epx` which has been previously run on the current directory, and reports message about test case validation.

5.8 Comparing

The command may be used to compare the local version of a file on the current directory with the one contained in the appropriate EUROPLEXUS directory. The file types that can be compared are : source (Fortran) files, include files, benchmark files, manual files, listing files, PostScript files or perl (utility) files. Multiple files may be compared (but all of the same type) by a single command.

The syntax reads:

```
epx_diff [-i] [-s] [-b] [-m] [-c] [-l] [-p] [-u] [-v] [-w]
[-a] [-g] [-f] [-?] [-d dir] [-t] file(s)[.<ext>]
```

where:

`-i` Option to compare an include file (extension `.inc`), rather than a source file (extension `.ff`). By default, a source file is compared.

`-s`	Compare files side-by-side, by highlighting the differences (this requires the text window to be at least 161 characters wide). By default, only the list of differences is printed.
`-b`	Option to compare a benchmark file (extension `.epx`), rather than a source file (extension `.ff`). By default, a source file is compared.
`-m`	Option to compare a manual file (extension `.ttx`), rather than a source file (extension `.ff`). By default, a source file is compared.
`-c`	Ignore case: uppercase and lowercase letters are considered identical.
`-l`	Option to compare a listing file (extension `.listing`), rather than a source file (extension `.ff`). By default, a source file is compared. To compare multiple listing files at one time, specify multiple names (e.g. by wildcards).
`-p`	Option to compare a PostScript file (extension `.ps`), rather than a source file (extension `.ff`). The reference file is taken from directory `%EUROPLEXUS%\Bench`. By default, a source file is compared.
`-u`	Option to compare a utility file (a Perl procedure, with extension `.pl`), rather than a source file (extension `.ff`). The reference file is taken from directory `%EUROPLEXUS%\Bench`. By default, a source file is compared.
`-v`	Option to compare a validation file (a text file, with extension `.vld`), rather than a source file (extension `.ff`). The reference file is taken from directory `%EUROPLEXUS%\Validate`. By default, a source file is compared.
`-w`	Option to ignore all blank space in the comparison.
`-a`	Option to compare an animation file (extension `.avi`), rather than a source file (extension `.ff`). By default, a source file is compared.
`-g`	Option to compare a graphics file (extension `.bmp`), rather than a source file (extension `.ff`). By default, a source file is compared.
`-f`	Option to display only the files with differences.
`-?`	Prints short help with purpose and syntax of this command.
`-d dir`	Specify a directory `dir` from which the reference files are to be taken, instead of having it determined by the file type. If used, this switch must be used after the switches that determine the file type (`-i`, `-b`, `-m`, `-l`, `-p`, `-u`).
`-t`	Same as `-l` (compare listings) but the different lines containing the date and time are not displayed. This reduces the size of the differences and makes easier to find the real differences.
`file`	Name of the file to be compared. No extension is required. If the extension is given, it must match the setting of the prevous option(s) if any.

Examples:

```
epx_diff celem
```

Compares the file `celem.ff` in the current directory with that from the EUROPLEXUS directory: `%EUROPLEXUS%\source\celem.ff`. Lists only the differences, with line numbers.

```
epx_diff -i contro
```

Compares the file `contro.inc` in the current directory with that from the EUROPLEXUS directory: `%EUROPLEXUS%\include\CONTRO.INC`. Lists only the differences, with line numbers.

```
epx_diff -s celem.ff
```

Compares the file `celem.ff` in the current directory with that from the EUROPLEXUS directory: `%EUROPLEXUS%\source\celem.ff`. Lists the files side-by-side and highlights the differences in the central (81-th) column.

```
epx_diff -l bm_str_eled01
```

Compares the file `bm_str_eled01.listing` in the current directory with that from the EUROPLEXUS directory: `%EUROPLEXUS%\bench\bm_str_eled01.listing`. Lists only the differences, with line numbers.

```
epx_diff -l *.listing >diffs.txt
```

Compares all listing files in the current directory with the corresponding ones from the EUROPLEXUS directory: `%EUROPLEXUS%\bench`. The resulting differences are written on file `diffs.txt` rather than on standard output, for easier inspection.

```
epx_diff -l –d bench2 *.listing >diffs.txt
```

Same as above example, but reference listing files are to be taken from directory `E:\EUROPLEXUS\Bench2` instead of the default directory (for listing files) `E:\EUROPLEXUS\Bench`.

The `epx_newer` command may be used to compare the time stamp (header) of the local version of source files on the current directory with the ones contained in the appropriate EUROPLEXUS directory. The file types that can be compared are : source (Fortran) files, benchmark files and manual files. All local files of the given type are compared with just one command.

This command may be useful for example if a user has copied some source files (e.g. by `epx_get`) onto a local directory without locking these files via the EUROPLEXUS development center. Perhaps some of the local files have been modified to produce a local test version, and some others have not. Then, after some time has passed, the user wants to submit an evolution containing the local modifications. He needs to know whether any of the files have been evolved in the meantime, i.e. whether newer versions of these files are available. For all such files, he will have to replace the modified files by the newer version and apply all modifications to this version.

The syntax reads:

```
epx_newer [-b] [-m] [-v] [-?]
```

where:

`-b`	Option to compare time stamp of benchmark files (extension `.epx`), rather than source files (extension `.ff`). By default, source file are compared.
`-m`	Option to compare time stamp of manual files (extension `.ttx`), rather than source files (extension `.ff`). By default, source filse are compared.

`-v` Option to compare time stamp of validation files (extension `.vld`), rather than source files (extension `.ff`). By default, source filse are compared.

`-?` Prints short help with purpose and syntax of this command.

5.9 Searching

The `epx_grep` command may be used to search all occurrences of a given pattern in the EUROPLEXUS source files. Either source (Fortran) files or include files or benchmark files or manual files or utility files may be searched.

The syntax reads:

```
epx_grep [-i] [-l] [-b] [-m] [-u] [-h] [-g] [-?] ["]pattern["]
```

where:

`-i` Option to search upon all include files (extension `.inc`), rather than upon all source files (extension `.ff`). By default, all source files are searched upon.

`-l` Option to print only the file names of the files containing the pattern. Each file is listed only once. By default, the file names, line number, and line contents (possibly multiple lines per file) are printed.

`-b` Option to search upon all benchmark files (extension `.epx`), rather than upon all source files (extension `.ff`). By default, all source files are searched upon.

`-m` Option to search upon all manual files (extension `.ttx`), rather than upon all source files (extension `.ff`). By default, all source files are searched upon.

`-u` Option to search upon all utility files (extension `.pl`), rather than upon all source files (extension `.ff`). By default, all source files are searched upon.

`-h` Option to search upon all history files (extension `.his`), rather than upon all source files (extension `.ff`). By default, all source files are searched upon.

`-g` Option to get also the concerned files.

`-?` Prints short help with purpose and syntax of this command.

`pattern` Pattern (string) to be searched. The search is case-insensitive and reports all found occurrences, with file name and line number. If the pattern contains blanks, it must be enclosed in double quotes.

Examples:

```
epx_grep rezoning
```

Searches all source files from the EUROPLEXUS directory for the string '`rezoning`'.

```
epx_grep -i IPB
```

Searches all include files from the EUROPLEXUS directory for the string '`ipb`'.

```
epx_grep "ipb ="
```

Searches all source files from the EUROPLEXUS directory for the string '`ipb =`'. Double quotes must be used because of the blank appearing in the pattern.

5.9.1 Searching module dependencies

When modifying a F90 module file, it is often important to know which files (subroutines, functions, or other module files) use, directly or indirectly, the module file under consideration. Usage of a module `m_xxx` may in fact be either direct, as in:

```
SUBROUTINE SUB1
USE m_xxx
…
```

or indirect, via nested modules, as in:

```
MODULE m_yyy
USE m_xxx
…
```

In the second case in fact, all files using module `m_yyy` also use `m_xxx`, and so on recursively.

The `epx_modtree` command may be used to (recursively) search all files that use a certain module, both directly and indirectly.

The syntax reads:

```
epx_modtree m_name
```

where:

`m_name` Name of the module, with or without the extension `.ff`.

This command produces a list of all (direct and indirect) users of the specified module in a local text file `_users.txt`. The users names are listed each in a separate line, in alphabelical order.

A related command, which searches the correct compilation order for an arbitrary set of source files, is the `epx_ordo` command described below in Section 6.15.

Another related command is `epx_get_users`. This command should be issued after epx_modtree: it reads the `_users.txt` file in the current directory and retrieves the standard verion of all the users files (except those which are already present in the current directory).

The syntax is simply:

```
epx_get_users
```

5.10 Filtering

The `epx_fil` command may be used to filter one or more files by the `epx_filter` utility described above in Section 5.1. Any type of text (ASCII) file may be filtered, including of course Fortran source and include files, C files and manual source (LaTeX) files. This command should be rarely needed, as the `epx_filter` utility is automatically invoked internally by the other commands (e.g. by the `epx_cmp` compilation command) as needed.

The syntax reads:

```
epx_fil [-i<ext>] [-o<ext>] [-c[<char>]] [-p<pwd> ...] file(s)
```

where:

`-i<ext>` Option to set the input file(s) extension to `.ext`. By default, the extension is `.ff`.

`-o<ext>` Option to set the output file(s) extension to `.ext`. By default, the extension is `.f`.

`-c[<char>]` This option, if present, is passed unchanged to `epx_filter` (see Section 5.1). It may be used to set the comment character to `<char>` rather than the default value `C`. If `<char>` is omitted, then in `epx_filter` neither the keyword lines nor the built-in message are transcribed, so the output file will contain just the activated text.

`-p<pwd>` Introduces a password (keyword) `pwd` to be passed to `epx_filter`. It may be repeated, and in that case the various passwords are concatenated (with a blank between words) before being passed to `epx_filter`.

`file(s)` Name of the file (or files) to be filtered, with extension. Note that wildcards may be used to specify multiple file names.

Examples:

```
epx_fil -pWIN celem.ff
```

Filters the source file `celem.ff` from the current directory by using the keyword `WIN`. The resulting file will be `celem.f`. The commented conditionals, if any, will start with the default comment character `C`.

```
epx_fil -iii -oinc -pWIN -pOGL *.ii
```

Filters all files `*.ii` from the current directory by using the two keywords `WIN OGL`. The resulting files will be `*.inc`. The commented conditionals, if any, will start with the default comment character `C`.

```
epx_fil -ittx -otex -c% gba_0010.ttx
```

Filters the manual LaTeX source file `gba_0010.ttx` from the current directory by using no keywords. The resulting file will be `gba_0010.tex`. The commented conditionals, if any, will start with the comment character `%`. The effect of using no keywords is that any conditionals will be de-activated.

5.10.1 Filtering a manual source file

A specific command has been prepared to filter a source file of the manual. The command reads:

```
epx_filter_manual [-q] [-k key] [-x] name(s)[.ttx]
```

where:

`-q` Quiet: do not echo (following) command line switches. Useful when the command is invoked from within a procedure.

`-k key` Add filtering key `key` to default filtering keys. This switch may be repeated to add as many keys as needed.

`-x` Suppress all default filtering keys (`WIN OGL`). This is sometimes useful in combination with the `-k` switch to get full control on the set of keys to be used during the filtering process.

Examples:

```
epx_filter_manual gba_0010
```

Filters the manual source file `gba_0010.ttx` from the current directory by using the standard keywords (`WIN OGL`). The resulting file will be `gba_0010.tex`. The commented conditionals, if any, will start with the comment character `%`.

5.10.2 Filtering a PostScript file

A specific command has been prepared to filter a PostScript file issued from EUROPLEXUS. The command reads:

```
epx_correct_ps [-t THICK] [-s SIZE] name(s)[.ps]
```

where:

`-t THICK` Use line thickness `THICK` (in points) instead of the EUROPLEXUS standard value which is 0.1 points. If not specified, a line thickness of 1.0 points is automatically set by the present command.

`-s SIZE` Use font size `SIZE` (in points) instead of the EUROPLEXUS standard value which is 9 points. If not specified, a font size of 10 points is automatically set by the present command, which also uses a bold font instead of the standard normal font.

The main use of this command is perhaps to increase font size and line thickness prior to transformation of the graphs into bitmaps to be included in publications, The default lines and fonts generated by EUROPLEXUS are too small for this purpose and result almost illegible, especially when they are largely reduced.

Examples:

```
epx_correct_ps -s 12 mytest
```

Filters the PostScript file `mytest.ps` by setting line thickness to 1.0 points instead of 0.1 points and font size to bold and 12 points instead of 9 points normal font.

5.11 Visualizing the manual

The `epx_manual` command visualizes (a copy of) the user's manual on the screen, in PDF format.

The syntax reads:

```
epx_manual
```

6 Procedures for automatic code evolution

As anticipated in Section 1, one of the most innovative characteristics of the EUROPLEXUS project with respect to its ancestors is the multi-site synchronized development. The code development is distributed among several “mirror sites”, of which one is defined as the “central mirror site”, see [3].

All sites contain, at any given instant in time, the same version of the code: this is the current EUROPLEXUS version. Developments are prepared and tested locally at the various development sites, and when they are ready for being included in the current version they are submitted to the central mirror site, which takes care of version “evolution”. The changes (“evolution sets”) are then distributed from the central site to all mirror sites for version synchronisation.

Each mirror site must therefore set up appropriate procedures for continuously updating the mirror code version stored locally. This is done by using the evolution sets received from the central site. Normally evolutions take place at the central mirror site not more than once a day, typically in the evening. The mirror code versions are also updated during the night at the various mirror sites so that users are not disturbed in their everyday work.

The following subsections describe the evolution strategy that has been adopted at the JRC site, and review the automatic procedures that have been set up to ensure smooth synchronisation of the mirror code version. First, however, a list and definition of the possible evolution sets is provided.

6.1 Evolution sets, packages and lists

Each "evolution set" received from the central mirror site may contain up to five types of evolution packages:

- Sources package: contains modified and new source files (`.ff`);
- Includes package: contains modified and new include files (`.inc`);
- Benchmarks package: contains modified and new benchmark test files (`bm_*.epx`, `bm_*.msh`, `bm_*.ps`, `bm_*.zip`, etc.). Note that each benchmark file `bm_xxx.epx` may optionally have an associated "zipped" package `bm_xxx.zip` containing any auxiliary files that are useful for the benchmark test, for example a mesh generation file or post-processing files for CASTEM 2000 (`bm_xxx.dgibi`) or other pre/postprocessors. If an associated mesh file `*.msh` exists, it must also be contained in the `.zip` package, besides being available as a separate file. Auxiliary files should preferably be pure ASCII files (not binary) for portability issues. However, the system does not check the contents of such `.zip` files (they are simply copied to the benchmarks directory) and the users are therefore responsible for this aspect. Such `.zip` files may be submitted during an evolution together with the associated `.epx` file: they are automatically propagated towards the mirror sites within the evolution's benchmarks package.
- Manuals package: contains modified and new manual files (`.ttx`).
- Validations package: contains modified and new validation files (`vl_*.vld`, `vl_*.zip`, etc.).

Note that an evolution may contain any or all of these packages, i.e. from 0 to 5 packages. The case of no packages corresponds to the null evolution, whose only effect is that of incrementing the evolution number contained in the evolution version file, see Section 6.3.3.

Packages are tarred, gzipped files, i.e. files that have been obtained by the `tar` Unix command (a port under Windows is also available, see 2.4) followed by compression through the `gzip` command. Compression is used to enhance the efficiency of automatic file transmission to mirror sites via `ftp`.

In addition to packages, each evolution set consists of two auxiliary text (pure ASCII) evolution lists:

- An evolution "resume" list, containing the list of the packages (if any) forming the evolution set;
- A "trace" list, containing the log of the evolution performed at the central mirror site.

These two lists are mandatory and must therefore be received with each evolution set.

6.1.1 Automatic distribution

Evolution sets are automatically transmitted via `ftp` during the night from the central mirror site to the mirror sites and placed in a special directory. This directory is protected and only accessible via a password, for obvious reasons of security.

The address of the JRC site, when accessed from outside, is currently ftp://elsa.jrc.it (IP address: 139.191.131.230), and the log-in userid is europlexus (a password is required). This corresponds locally to (protected) directory: `\\sm48\E:\EUROPLEXUS\Iis\ftp\europlexus`. The directory is monitored by the automatic code evolution procedure, which is started each night at a pre-set time (see Section 6.3.1).

6.1.2 Naming conventions

The naming conventions for the evolution packages and the evolution lists is summarized in Table 4 and obeys to the following rules:

- Evolution lists have the extension `.txt` while evolution packages have the extension `.tar.gz`.
- Each list or package name starts with a fixed (mnemonic) letter, followd by an underscore: `R_` for resume list, `T_` for trace list, `S_` for sources package, `I_` for includes package, `B_` for benchmarks package, `M_` for manuals package and `V_` for validations package.
- The rest of the list or package name consists of a 12-character string, of the form: `NNNNxDDMMMYY`, where `NNNN` is the evolution number, for evolution lists, or the evolution index, for evolution packages, `x` is the lower-case letter x (fixed separator), `DD` is the day of the month in digits, `MMM` the abbreviated month name (in characters, of which the first one is uppercase), and `YY` the last two digits of the year.
- Numeric fields such as `NNNN`, `DD` and `YY` contain leading zeroes as needed (for example the `NNNN` field corresponding to evolution number 12 is `0012`).

Table 4 - Naming conventions for evolution lists and packages

Name example	File/Package type
`R_0012x03Feb00.txt`	Resume file
`T_0012x03Feb00.txt`	Trace file
`S_0009x03Feb00.tar.gz`	Sources package
`I_0002x24Jan00.tar.gz`	Includes package
`B_0004x28Jan00.tar.gz`	Benchmarks package
`M_0001x01Feb00.tar.gz`	Manuals package
`V_0001x28Jan05.tar.gz`	Validations package

6.1.3 Numbering conventions

It is important to note that, while all lists and packages belonging to the same evolution set are characterised by the same date, and therefore the field `DDMMMYY` is the same, this is not true for the `NNNN` field, because of its different meaning for evolution lists or packages.

The precise definition is as follows:

- The evolution number is the number of the evolution set, and is therefore incremented by one at each evolution (even for null evolutions). There is therefore a one-to-one correspondence between evolution number and the date field.
- The evolution index is related to the package type. It is also incremented by one at each evolution, but only if the corresponding package is present in the evolution. This is to avoid the presence of empty evolution packages.

As a consequence of the above numbering rules, an evolution typically contains the following:

- All lists and packages have the same date field: the date of the evolution.
- The two evolution lists have an `NNNN` field corresponding to the evolution number (the same for both lists, then).

- The evolution packages belonging to a same evolution set have in general different evolution indexes. Each index is lower or equal to the evolution number, and corresponds to the number of times the corresponding package type has been effectively present in all evolutions so far.

6.1.4 History files

In addition to the evolution lists and packages presented in Table 4, there exists also another type of information sent with the evolution: history files. These are not sent separately, but are contained (`tar`-red) within the corresponding evolution packages (sources, benchmarks, manuals). Note that there exists no history file relative to includes.

The name of an history file is the same as that of the corresponding package file, except that:

- The two initial characters are replaced: `S_` is replaced by `Cs` for source histories, `B_` is replaced by `Cb` for benchmark histories, `M_` is replaced by `Cm` for manuals history and `V_` is replaced by `Cv` for validations history.
- The file name extension is `.hist` instead of `.tar.gz`.

Examples are given in Table 5.

Table 5 - Naming conventions for history files

Name example	Corresponding package	History file type
`Cs0009x03Feb00.hist`	`S_0009x03Feb00.tar.gz`	Sources history
`Cb0004x28Jan00.hist`	`B_0004x28Jan00.tar.gz`	Benchmarks history
`Cm0001x01Feb00.hist`	`M_0001x01Feb00.tar.gz`	Manuals history
`Cv0001x28Jan05.hist`	`V_0001x28Jan05.tar.gz`	Validations history

An history file contains the information relative to the evolution of the corresponding file types (e.g. sources) contained in the evolution. For each evolved file, the file header followed by the comments typed by the developer who has evolved the file are contained in the history file. Histories relative to different files are separated by a blank line.

The structure of a typical history file is then as follows:

```
== Evolution EUROPLEXUS #0009 du 00/01/28 a 22:00:01 : benchs  #0004 ==

$ BM_CIR_CABIF               TOUS JRC       00/01/28 22:00:11
Changed tolerance of first QUAL test (VITE 1 1) because
it did not pass under NT. From 1.E-3 to 4.E-3.

$ BM_CIR_TCHOC               TOUS MLEPAREUX 00/01/28 22:00:13
Ajout de la directive QUALIFICATION.

$ BM_CIR_TUYPL               TOUS MLEPAREUX 00/01/28 22:00:15
Ajout de la directive QUALIFICATION.

================================================================================
```

The first line contains information about the evolution number (`#0009` in the above example), date (`00/01/28`) and time (`22:00:01`), followed by the type of history file (`benchs`, i.e. benchmarks in this case; other possible types are `sources`, `manual` or `validation`) and the evolution index for this type of package (`#0004`).

Then follows a blank line. Next we have for each evoluted file the file header (one line) followed by the file comments (any number of non-blank lines), followed by a blank line. The history file is terminated by a line containing a series of '='.

If the evolution set contains some new or modified includes file (package `I_`), the history file for these include files is not foreseen at the central mirror site (as includes are more or less "frozen" in EUROPLEXUS and will gradually be replaced by modules) and therefore there exists no includes history file. However, the list of the modified includes (one per line) appears in the history file relative to the sources (if any) of the same evolution set, directly after the first line and before any header.

6.2 Evolution strategy

The evolution procedure contains the following steps:

1. Produce a modified load module taking into account changes and additions to sources and includes, if necessary (i.e., if either the sources package, or the includes package, or both are present);

2. Run the benchmark tests suite: the test files in the benchmark package, if any, have the precedence over the homonyms from the benchmarks directory;

3. Evolve the sources (if any), includes (if any), benchmarks (if any), validations (if any); update the objects library and the load module;

4. Evolve the manuals, if any;

5. Evolve the history files from the global history files contained in the sources, benchmarks, manuals or validations package (if any).

The following observations apply:

- The manuals package, if present, is practically independent from the other packages. We therefore process it after all other packages.

- The benchmarks package, if any, applies only to the testing phase, which needs a load module. Therefore, its processing must occur after that of sources and includes, if any.

- Evolution of the validations package, if present, consists simply in copying the received files `vl_*.vld` and `vl_*.zip` to the `validate` sub-directory and in updating the corresponding history files. These examples are not exectuted automatically by the system, unlike the non-regression benchmark tests.

- The sources and includes packages are in general inter-related, and care must be taken as to the order in which operations are performed.

For the treatment of the sources and includes packages, the following strategy is applied:

- If only the sources package is present (no includes), then it is treated first, followed by benchmarks, followed by sources and benchmarks evolutions, followed by manuals treatment and evolution, and finally by histories evolution, if any.

- If only the includes package is present (no sources), then it is treated first, followed by benchmarks, followed by benchmarks evolution, followed by manuals treatment and evolution, and finally by histories evolution, if any.

- If both sources and includes packages are present, then the new or modified sources might need the new or modified includes in order to be compiled correctly. Therefore, we proceed in the following order:

    1. Unpack the includes package, so the new or modified `.inc` files become available;

    2. Treat (i.e., unpack and compile) the sources package;

3. Find all files which need any of the new or modified includes, but are not among the modified sources. Retrieve them, compile them, then delete the sources (only the corresponding `.obj` and `.mod` files will be evoluted, since the source has not changed).

4. Proceed with benchmarks, followed by sources, then by manuals, and finally by histories evolution, if any.

6.3 Evolution driver procedure

Code evolution is driven by the perl script `epx_evol_start`. This procedure is automatically started each night as described in Section 6.3.1 and verifies whether one or more new evolution sets have been received from the central mirror site in the `ftp` directory (see Section 6.1.1).

If this is the case, it performs one or more evolutions of the local mirror code version, by calling several lower-level perl scripts in the following order:

- Script `epx_test_sources` for the testing of sources;
- Script `epx_test_includes` for the testing of includes;
- Script `epx_lk` for producing a local executable;
- Script `epx_test_benchmarks` for the testing of benchmarks;
- Script `epx_test_manuals` for the testing of manuals;
- Script `epx_evol_histories` for the evolution of histories.

All these scripts are perl procedures that are transformed into batch files, according to the technique explained in Section 3.2.1, so as to allow for output redirection. The driver procedure `epx_evol_start`, the histories evolution procedure `epx_evol_histories`, the mail sending script `epx_mail`, the obsolete file removal script `epx_obsolete` (see Section 6.10) and the source backup script `epx_save` (see Section 6.11) may only be executed by the local site administrator, for obvious security reasons. The `epx_setat` command, described at the end of Section 6.3.1, may only be executed by the administrator because the `at` command requires administrator priviledges.

However, all other procedures may be executed also by normal users, since they are useful to test local developments before submitting them to the central site for evolution. These procedures are harmless, because all changes to the current mirror version are performed exclusively by the driver procedure and by the histories evolution procedure.

The various scripts are described below in Sections 6.4 to 6.8 in the order in which they are called by the driver procedure. All procedures are listed in the Appendix, and are summarized in Table 6 for ease of reference. The names of procedures that may be invoked only by the site administrator are enclosed in brackets.

Table 6 - Resume of commands for version testing and evolution

Scope	Syntax	Section
Driving code evolution	`[epx_evol_start [-p] [-noqw] [-nompi] [-nocheck] [-fullcmp]]`	6.3
Testing all benchs with -check	`[epx_evol_check]`	10.15.4
Testing sources	`epx_test_sources`	6.4
Testing includes	`epx_test_includes`	6.4
Producing batch executable	`epx_mkbatch`	6.5.1
Testing benchmarks	`epx_test_benchmarks [-e <exec>[.exe]] [-l] [-s]`	6.5

Scope	Syntax	Section
Testing manuals	`epx_test_manuals`	6.7
Correcting accents	`epx_accents name(s)[.ttx]`	7.6
Testing a single manual page	`epx_test_man name[.ttx]`	7.7
Manuals pass 1	`epx_pass_1`	7.5.2
Manuals pass 2	`epx_pass_2`	7.5.3
Manuals pass 3	`epx_pass_3`	7.5.4
Manuals pass 4	`epx_pass_4`	7.5.5
PDF manuals pass 1	`epx_pass_1_pdf`	7.5.6
PDF manuals pass 1	`epx_pass_1_pdf`	7.5.7
PDF manuals pass 1	`epx_pass_1_pdf`	7.5.8
Cleanup manual files	`epx_clean_manual_files`	7.5.2
Testing and evolving histories	`[epx_evol_histories]`	6.8
Sending e-mail to partners	`[epx_mail status num filename]`	3.5, 6.9
Setting aside obsolete files	`[epx_obsolete [-i] [-b] [-m] name[.ext]]`	6.10
Backup complete code source	`[epx_save]`	6.11
Restore complete code source	`[epx_restore]`	6.12
Compiling system from scratch	`[epx_make]`	6.13
Set at commands	`[epx_setat]`	6.3.1
Prepare for batch evolution	`epx_make_evo`	6.15
Prepare for batch evolution	`evototgz`	6.15
Test evolution package	`epx_test_evo`	6.15
Generating debuggable project	`epx_init [-w] [-e] [-d] [-?]`	8.3
Getting evolution packages	`epx_ftp_getfiles`	10.8.2
Updating on-line executable	`epx_ftp_putexe [-w] [-M] [-c]`	10.9
Updating on-line executable	`epx_ftp_putexe_64 [-M]`	10.16.6
Updating on-line manuals	`epx_ftp_putman`	10.9
Updating file header	`epx_update_header [-i] [-b] [-m] name(s)[.ext]`	6.16

6.3.1 Automatic startup of the evolution driver procedure

Automatic execution of the evolution driver procedure is accomplished by means of the `at` command. This command may either be used from the command line or as a GUI-based command (`winat.exe`) that can be found in the NT Resource Kit (see Section 2.4.1).

In order for the `at` or `winat` command to be available, the following two requirements must be satisfied:

- The Scheduler service must run on the machine (this is the case for the EUROPLEXUS site administration machine, smnt04 at JRC);
- The user must have System Administrator priviledges.

Once scheduled a command via `at` or `winat`, the command is automatically executed on the site administration machine with the chosen frequency (for example, each night at a certain time) even though nobody is logged in.

It is important to note that `at` or `winat` commands run as if they were issued by the SYSTEM user. Therefore, they "see" the environment of this user. For this reason, environment variable settings must be applied to System Variables and not to User Variables in the case of the site administration machine (see Sections 3.1.2 and 4.1.1).

The `at` command for the execution of the evolution driver procedure reads:

```
at 07:00 /every:M,T,W,Th,F,S,Su
                    "cmd /c E:\EUROPLEXUS\Util\epx_evol_start"
```

(on just one line), where `07:00` is the time at which the command must be executed, `/every:M,T,W,Th,F,S,Su` are the days of the week at which the command must be executed (every day in this case), and the string in double quotes is the command to be executed.

Note that, according to on-line help:

- The command interpreter (`cmd -c`) must be explicitly invoked because `at` does not load `cmd.exe` by default. Therefore, it may execute directly only `.exe` files, while `epx_evol_start` is a `.bat` file.
- The absolute path of the command, i.e. the entire pathname beginning with the drive letter must be given, hence we use `E:\EUROPLEXUS\Util\epx_evol_start` instead of just `evol_start`.
- No STDOUT or STDERR redirection is contained in the above command. This is because the `epx_evol_start` procedure does its own redirection internally, see Section 6.3.2 below. Output or error messages coming from the execution of this procedure are appended (>>) to, and may be found on, file: `E:\EUROPLEXUS\Fromcentral\evol.log`.

Typing `at` without arguments gives a list of currently scheduled commands, each identified by an index. Typing `at` followed by a command index, gives details on that command.

The `winat` command offers the same functionality as `at`, in a GUI interface. The use is straightforward, but note that the command must not be specified in double quotes in this case.

Under NT 4.0 Service Pack 4, it was discovered by chance that the short form of the command, and without explicitly invoking `cmd.exe`, also works. Therefore, it seems that the first two constraints listed above do not hold in reality. The command is currently entered simply as: `epx_evol_start`.

Under NT 4.0 Service Pack 6, the `at` command executed from the command line has a different behaviour:

- The command interpreter (`cmd -c`) must be explicitly invoked.
- The absolute path of the command, is not necessary.
- The double quotes around the command do not work.
- Redirection of command output does not work.

To overcome the latter malfunctioning, any output or error redirections, if necessary, must be performed within the command itself (i.e. within the batch script). An example of this is the `epx_save` script, see Section 6.11.

Under Windows 2000 Professional the same behaviour as under the latest NT release is observed.

The `at` commands necessary for EUROPLEXUS development are issued by means of the following command (batch file `epx_setat.bat`, see Section):

```
at 07:00 /every:M,T,W,Th,F,S,Su cmd /c epx_evol_start
```

```
at 11:00 /every:Su cmd /c epx_save
```

The first command performs the automatic code evolution every morning at 07:00, if an evolution package is available. The second command performs a full backup every Sunday morning.

6.3.2 Output and error redirection strategy for the driver procedure

The evolution driver procedure is so set up as to be executed via the `at` command. The output and possible error messages issued from this procedure are treated as follows:

- On procedure entry, the STDOUT and STDERR are internally redirected, via the `open` perl operator, to a fixed file ("global log file") `E:\EUROPLEXUS\Fromcentral\evol.log`, in append mode; it is therefore implicitly assumed that the path and file always exist and are accessible.

- When a suitable evolution package is found (see Section 6.3.4 below), and an actual evolution is started, STDOUT and STDERR are closed and re-opened to a file ("individual log file") with the name `%EUROPLEXUS%\Fromcentral\NNNN.log`, where `NNNN` is the evolution number (See Section 6.1.3). This file will then contain the log relative to this evolution only.

- If the evolution terminates correctly, STDOUT and STDERR are closed, and the log file relative to evolution `NNNN` is e-mailed to project partners. The file is also archived for later inspection.

- This process is repeated for as many evolution packages (in the correct order) as can be found on the `ftp` directory.

- Finally, the STDOUT and STDERR are re-attached to the global log file, and the closing messages are issued, reporting the total evolution(s) time.

- If an evolution produces an error, the error message is printed to the individual log file, then STDOUT and STDERR are closed and the individual log file is e-mailed to all project partners.

In summary, the global log file contains just a rough trace of (all) evolutions, while the detailed trace of each particular evolution may be found in the corresponding individual log file in the archive directory.

6.3.3 Evolution version file and evolution lock file

At the beginning of its execution, the `epx_evol_start` procedure verifies the existence of two auxiliary files:

- The evolution version file: `%EUROPLEXUS%\VERSION.txt`;

- The evolution lock file: `%EUROPLEXUS%\evolution.lck`.

The evolution version file is a text file containing just one line of text with the number of the current local mirror version of EUROPLEXUS. This number coincides with the evolution number (see Section 6.1.3) of the last evolution that has been successfully applied. The number is updated (incremented by one) at the very end of the evolution process, each time a new evolution is successfully applied.

The evolution lock file is also a text file. Its contents is irrelevant, but its presence indicates that the evolution process is locked. If such a file is found to exist at the start of the evolution process, the procedure `epx_evol_start` immediately exits with an appropriate message.

Else, the procedure starts its work and, if it finds an appropriate evolution set (i.e. one with an evolution number exactly equal to the number contained in `VERSION.txt` plus one), it starts the real evolution process by creating a lock file.

If an error is encountered during the evolution, the procedure exits and the lock file thus remains. This precludes the application of further evolutions (in the following nights) as long as the problem has not been removed by manual intervention of the mirror site responsible.

If the evolution terminates correctly, the driver procedure removes the lock file after incrementing by one the evolution number in the version file.

In case of errors in the automatic evolution during an absence of the site administrator (e.g. holidays period), the evolution sets coming from the central mirror site are cumulated in the local `ftp` directory. As soon as the problem is corrected and the evolution lock file is manually removed, the evolution driver procedure tries to apply all cumulated evolution sets one after the other, in the correct sequence.

6.3.4 Searching an appropriate evolution set

After the preliminary operations described in the previous Section, the evolution driver procedure searches the `ftp` directory for an appropriate evolution set. The process is iterative, so that if several evolution sets are present in the directory, they are all applied, one after the other, in the correct order (provided of course no errors occur during the evolutions).

The presence of a set is recognized by the existence of a file whose name matches that of an evolution resume list: `R_NNNNxDDMMMYY.txt` (see Section 6.1.2). When such a list is found, its evolution number `NNNN` is checked: if it corresponds to the value in `VERSION.txt` plus one, it is processed, else it is skipped.

Processing the resume list consists of opening it and reading the list of the evolution packages, if any. Recall (Section 6.1) that `0` to four packages may be listed. Then the procedure verifies the existence of the listed packages, if any, and of the (mandatory) evolution trace list `T_NNNNxDDMMMYY.txt`.

All evolution packages present in the set are copied from the `ftp` directory to the evolution directory. At JRC this directory is `%EUROPLEXUS%\Fromcentral`.

6.4 Testing includes and sources

The next step is the testing of modified or new include and source files, if any. As anticipated in Section 6.2, these two packages are closely related, because include files are needed in the compilation process of source files.

6.4.1 Making any new include files available

If an includes package is present, it is opened and untarred first, so that the new or modified include files become available in the evolution directory. Then the sources package, if any, is opened and untarred. The procedure verifies that an appropriate history file is contained in the sources package.

6.4.2 Compiling the modified source files

Next, the driver procedure calls the first lower-level perl script, `epx_test_sources`. This script behaves differently depending on whether or not a file called `epx_ordo.txt` is present in the current directory. See Section 6.14 to learn how to produce this file if necessary.

Presence of the epx_ordo.txt file

If present, the file `epx_ordo.txt` should contain a list of the names of the source files to be compiled, one per line, in the order in which they must be compiled. Recall, in fact, that in F90 the order of compilations is relevant, e.g. in the presence of (possibly nested) modules. Normally, a file `epx_ordo.txt` is generated at the central mirror site and is included in the sources package with each evolution of the sources. When such a file is found in the current directory, the `epx_test_sources` script compiles only the sources listed in this file, and in the order given. If a file is listed, but not present, an error is raised. However, if a file is present but not listed, it is simply not compiled.

Absence of the epx_ordo.txt file

When, on the other hand, no file `epx_ordo.txt` is found in the current directory, then the `epx_test_sources` script adopts the following strategy.

The version of the script available until end April 2003, compiled all source files present in the evolution directory three times:

- The first time, only module files (whose name matches the pattern `m_*.ff`) are compiled by the `epx_cmp` command (Section 5.4). Errors may occur in such compilations, because of module inter-dependency, and are therefore tolerated at this level.
- The second time, only module files are compiled once more. Errors are again tolerated.
- The third time, all source files are compiled, and in this phase errors are not tolerated.

This way of proceeding ensures proper compilation up to a level 2 of nesting between modules (i.e. a "first level" F90 module may `USE` a "second level" module). This is sufficient for the current version of the code, but in general deeper module nesting may occur (the maximum allowable nesting level may depend upon the compiler).

It is obvious that, in the absence of the `epx_ordo.txt` file, the present strategy is empirical and should be ameliorated in the future (a valid alternative is to produce the order file, see Section 6.14). Note that the GUI compilation environment (Developer Studio, see Section 2.2) automatically finds module inter-dependencies and compiles the various components of a FORTRAN project in the correct order. But a similar facility is not available for command-line driven compilations.

Starting in May 2003, the script generates itself a file `epx_ordo.txt` by invoking the `epx_ordo` procedure described in Section 6.14. It then proceeds like in the case that the file would be present in the distributed evolution package.

6.4.3 Compiling the "includers" source files

Next, assuming that no (fatal) compilation errors have occurred, if there exists an includes package the driver procedure calls the lower-level perl script, `epx_test_includes`. Note that the includes package has been already untarred by the previous script, so this script starts by searching (by means of the `epx_grep` command decribed in Section 5.9), the names of all source files that use (`INCLUDE`) any of the `*.inc` files present in the evolution directory. This is called the "list of includers".

The includers list is sorted by removing multiple occurrences (`sort -u` command) and then a loop over all file names is performed:

- If the current file is already present in the evolution directory, it has been already compiled by the previous script, therefore it is skipped;
- Else the file is retrieved by means of the `epx_get` command (Section 5.3), compiled, and finally removed from the evolution directory. In this way only the object (`.obj`) file and possibly the binary module (`.mod`) file corresponding to the retrieved source remain in the evolution directory and will be updated in the last phase of the evolution. There is in fact no need to update the source (`.ff`) in this case.

Note that the compilation process performed by this script follows exactly the same triple compilation scheme described above for the "old" sources evolution script (version before May 2003) when no `epx_ordo.txt` file is available: modules only (by accepting errors) are treated the first two times, all files the third time (no errors tolerated).

6.4.4 Generating the local executable

The next step is then the production of a local executable file (`epx.exe`) containing all the modifications introduced in either the include or the sources, or both. This is accomplished by the linking command `epx_lk` described in Section 5.5.

Note that, more precisely, the local load module is produced not only if either a sources or an includes or both packages are present, but also in the case that none of these are present but there exists a benchmarks evolution package. In this latter case, the link procedure will automatically extract the `main.obj` file from the EUROPLEXUS object library, perform the link and then erase this file (see Section 5.5). The produced local executable will then be virtually identical to the current standard EUROPLEXUS executable. This may seem redundant but it would be not, for example, in case a specialised batch version of the code would be needed to execute benchmark tests.

6.5 Testing non-regression benchmarks

The next step concerns executing the standard, non-regression benchmark tests. This phase is performed if any of the three packages: sources, includes, benchmarks, is present. In the first two cases, execution is necessary because the code has changed, while in the third one it is necessary because the tests themselves have changed.

First, the benchmarks evolution package, if any, is opened and untarred (by verifying that it contains a proper history file). Next, the benchmarks suite is executed, by calling the lower-level script `epx_test_benchmarks`.

This script proceeds as follows:

- First, a loop is performed that executes all benchmark tests, if any, present in the evolution directory, via the "`epx_bench -l -b`" command (Section 5.6). These are the new or modified tests contained in the benchmarks evolution package.

- Next, a second loop is performed over all benchmark tests (`.epx` files) contained in the EUROPLEXUS `bench` directory via the "`epx_bench -b`" command, but only in case that either the sources or the includes have changed (this second loop is not performed if only the benchmarks have changed). If a benchmark input file with the same name is present in the evolution directory (and therefore the benchmark has been already executed during the first loop), then the case is skipped. Else it is copied to the evolution directory and executed by means of the `epx_bench` command (now without the "`-l`" switch).

At the end of each benchmark execution, the `epx_bench` command verifies the test results by internally calling the `epx_vali` script. All benchmark tests are executed, even in the case that validation failures are encountered in some of the tests. In case of test validation failures, the procedure stops only at the end of the benchmarks tests by reporting the number of failed tests.

The command syntax is as follows:

```
epx_test_benchmarks [-e <exec>[.exe]] [-l] [-s] [-b]
```

where:

`-e`	Use executable `<exec>[.exe]`, not local one (`epx.exe` by default).
`-l`	Perform local tests only, not local tests plus standard tests.
`-s`	Use standard executable, not local one (`epx.exe` by default).

6.5.1 Executing benchmarks in batch mode

Note that the following discussion, and the `epx_mkbatch` command, have become obsolete in May 2003 with the introduction of OpenGL graphics and the use of a console-based standard executable, see Section 10.10.2. It has been maintained in the present report because it might be useful for other mirror sites that adopt a different type of standard EUROPLEXUS executable.

When executing the benchmarks suite by the `epx_test_benchmarks` command, the local executable `epx.exe` is used by default. This executable, normally produced by the "`epx_lk -w`"

command, is a QuickWin application so that interactive graphics may be produced if necessary in order to test new developments or modifications. When the executable is launched, it opens a window on the screen, known as the QuickWin frame window.

However, when the benchmarks suite is executed, which consists of over 150 test cases driven from the `epx_test_benchmarks` script, the presence of the frame window may be disturbing. In fact, in this case the code does not need any interactive user intervention, therefore the presence of this window is redundant. This is not usually a problem during automatic code evolution, which takes place early in the morning and in unattended mode (no user is logged in the machie at that time, usually). But it may be quite disturbing when executing the benchmarks suite during the day, e.g. right before submitting a set of files for evolution. In fact, since the tests suite takes about ten minutes (on a Pentium 1.5GHz) to complete, the machine remains almost unusable for other (interactive) purposes during that period.

In order to circumvent this problem, it is possible to produce a special version of the executable, which is suitable for "batch" execution only, and does not produce a frame window. This version should clearly be used only for executing the benchmark tests suite. This may be obtained by the command:

```
epx_mkbatch
```

This command performs the following operations:

- It retrieves to the current directory the source file `ifwin.ff`, unless this file is not yet present in the current directory. This file contains a few subroutines specific for the Windows version of the code, and in particular for activation of the QuickWin interface. If the file is already present, the local version is used.

- It compiles the file with the command `epx_cmp -w -x -k WIN ifwin`: this filters the source file by using the `WIN` keyword, but not the `QWIN` keyword as would be by default. The result is that calls to the QuickWin initialisation routines (in particular, `CALL MWEXIT` and `CALL MWSETP`) are filtered out from the source. In this way, the application may be built as a "Console application" rather than as a "QuickWin application".

- It links and produces a local "console application" executable `epxb.exe`, by the command `epx_lk -o -f -e epxb.exe`. The name is different from the default (`epx.exe`) in order to distinguish it from a normal (QuickWin) executable that could be possibly present in the directory. Note the `-f` switch that is used in the link command: this forces an executable to be produced even though there are link errors. This switch is necessary because there remain a few missing symbols (QuickWin library routines) at link time, which are called from routines other than `ifwin.ff`. These errors could be avoided by filtering also these other routines in addition to `ifwin.ff`, but the process would become too complex. The present strategy is considered sufficient because in any way the "batch" module will be used only to execute the benchmarks suite.

- It removes the files `ifwin.*` from the current directory, if the file `ifwin.ff` was not originally present (in order to keep the directory clean).

Once produced the `epxb.exe` executable, this may be used to run the benchmarks suite by invoking the following command:

```
epx_test_benchmarks [-l] -b
```

6.6 Updating sources, includes, executable, benchmarks and validations

At this point of the evolution, actual updating of the packages that were (successfully) tested so far is performed by the driver procedure `epx_evol_start`. This is performed by replacing the source, include etc. files of the EUROPLEXUS directory with the new versions present in the evolution directory. The following file types are copied to the appropriate directories:

- Includes (`.inc`), if any;
- Sources (`.ff`), if any;
- Compiled modules (`.mod`), if any;
- Object files (`.obj`), if any, are replaced in the `libplex.lib` object library by means of the `lib` command;
- Executable load module (`epx.exe`), if any.
- Benchmark files (`.epx`, `.msh`, `bm_*.zip`, `.listing`, `.ps`), if any.
- Validation files (`vl_*.vld`, `vl_*.zip`), if any.

Note that each validation example present in the validations sub-directory must consist of two files:

- A description (text) file `vl_xxx.vld`, containing a description of the validation test. This is a pure ASCII file.
- The associated validation package file `vl_xxx.zip`, containing a set of files relative to the validation example (or examples). These should allow a generic user to reproduce the example calculation and verify the results, and are typically an input file, a mesh file, a document file in PDF or other format, output results in the form of curves (e.g. PostScript files) or iso-value plots, etc. The system never checks the contents of the `.zip` file.

6.7 Testing and updating the manuals

The next phase of the evolution is the testing and updating of manuals. This phase is independent from the previous ones and is performed only if a manuals evolution package is present in the evolution set.

The manual evolution package is opened and untarred, and the driver procedure verifies the existence of an appropriate history file.

Next, the manuals are tested by calling the lower-level script `epx_test_manuals`. If this is successful, the manual files are updated by copying to the EUROPLEXUS manual directories (`manual` or `manual_filtered`, as appropriate) the following files from the evolution directory:

- LaTeX source files (`.ttx` and `.tex`);
- LaTeX device-independent binary file (`manual.dvi`);
- PostScript file (`manual.ps`);
- Portable Document Format (PDF) file (`manual.pdf`);
- HTML file produced by the `tth` utility (`manual.html`);
- HTML “monolithic” file produced by the `hevea` utility (`manual_h.html`);
- HTML “split” files produced by the `hevea/hacha` utilities (the whole contents of the `hacha` subdirectory in the evolution directory).

The organisation of the EUROPLEXUS User’s manual and the detailed functioning of the `epx_test_manuals` script are described below in Section 0.

6.8 Testing and updating the history files

The next step of the evolution is the updating of history files, which is performed in case any type of global history file (sources, benchmarks or manuals, see Section 6.1.4) has been encountered in one of the preceding evolution phases.

The `epx_evol_histories` lower-level perl script is invoked repeatedly, each time passing the name of a global history file. The script parses the global file (`.hist`), extracts the header and comments relative to each evolved file, and appends them to the corresponding history file (`.his`) in the EUROPLEXUS `history` directory.

If the global history file is a sources history file, it may also contain the names of evolved include files, as described in Section 6.1.4. Since no evolution comments are available for these files, the procedure simply appends to their history files a line containing the evolution number, date and index (first line of the global history file, see Section 6.1.4).

6.9 Final evolution operations

If all preceding operations were successful, the evolution is terminated by the following actions:

- The evolution resume and trace lists, plus the evolution package files, if any, are copied to the `backup` directory, and then they are deleted from the `ftp` directory;
- The evolution number contained in file `VERSION.txt` is incremented by one;
- The evolution directory is cleaned up: all files are deleted, except the local evolution log file `evol.log`, which is copied to the `trace` directory;
- The evolution lock file `evolution.lck` is removed, thus rendering further evolutions possible;
- The local evolution log file `NNNN.log` is automatically e-mailed to all project partners by calling the `epx_mail` script, then the file is moved to the `trace` directory;
- The `ftp` directory is searched for the next evolution, if any.

Note that in case of errors between the start and end of the `NNNN`-th evolution, the evolution driver procedure stops without removing the lock file, but before doing that an error message containing also the evolution log file produced so far is automatically e-mailed to all project partners.

6.10 Dealing with obsolete files

It may happen from time to time that some files belonging to the EUROPLEXUS project become obsolete and should be removed from the system. In this case, the central site sends a message to all mirror sites containing the list of the files to be removed.

This operation may be risky if performed manually, therefore a command has been prepared. This command, called `epx_obsolete`, may only be invoked by the local site administrator.

The command may deal with source files, include files, benchmark files or manual files. Instead of removing the files, they are moved to a subdirectory `\obsolete` under the directory in which they are placed. Any previous obsolete files with the same names are overwritten.

The command syntax is as follows:

```
epx_obsolete [-i] [-b] [-m] [-v] name[.ext]
```

where:

`-i` The file to be treated is an include file, not a source file.

-b	The file to be treated is a bench file, not a source file.
-m	The file to be treated is a manual file, not a source file.
-v	The file to be treated is a validation file, not a source file.
name	Name of the file, with or without extension.

If an history file name.his exists, corresponding to the file name given, it is moved to the obsolete history directory.

If the file is a bench file, then all related auxiliary files name.* are also moved to the obsolete bench directory. These are for example the mesh file, the listing file, etc.

If the file is a validation file, then all related auxiliary files name.* are also moved to the obsolete validation directory. This is typically the associated (mandatory) zip file.

If the file is a source file, then in addition to the above treatment the corresponding object file name.obj is removed from the EUROPLEXUS object library file (libplex.lib). Furthermore, if the file is a module file (i.e. if a corresponding file name.mod exists in the EUROPLEXUS module subdirectory), then the corresponding name.mod file is also deleted.

6.11 Producing a complete backup of the system source

It is useful for various reasons to produce from time to time a complete backup of the system's source, including the Fortran source, include files, manual pages, benchmark tests, script procedures etc.

The command syntax is as follows:

```
epx_save
```

and may be executed only by the local site administrator. This command produces a tar file containing all the above mentioned system sources, compresses it and stores it in the directory E:\EUROPLEXUS\Save. The file name is of the form epx_nnnn.tar.gz where nnnn represents the current system version number. The command is scheduled to start every Sunday morning, so that normally one backup per week is produced.

Before producing the file, the script verifies that a file epx_nnnn.tar.gz is not already present in the Save directory. Therefore, if the system version has not changed from the previous backup, no new file is produced.

6.12 Restoring a previously saved complete backup

Under exceptional circumstances it might be useful to restore a complete backup of the system's source, previously produced by the epx_save command described above.

The command syntax is as follows:

```
epx_restore nnnn
```

and may be executed only by the local site administrator. This command reads and extracts a tar file epx_nnnn.tar.gz from the directory E:\EUROPLEXUS\Save. This file must have been previously produced by the epx_save command, and the nnnn string represents the system version number of the backup file.

Be warned that this file replaces the complete set of EUROPLEXUS sources, including the Fortran source, include files, manual pages, benchmark tests, script procedures etc. Therefore, it should be used only exceptionally, e.g. in the initial implementation of a new mirror site. Note furthermore that:

- This particular command, i.e. the `epx_restore.pl` and the corresponding `epx_restore.bat` files, may not reside in the `Util` subdirectory of the EUROPLEXUS mirror site, unlike all other commands presented in the present report. The reason is that, since the utility files themselves are backed up and restored by the `epx_save` and `epx_restore` commands, trying to update the `epx_restore` file itself while it is being used would produce an error under the Windows system.

- The `epx_restore` command just updates the various source files (Fortran, manual, benchmarks etc.) but it does not rebuild any object files, libraries, or the executable. To this end, the `epx_make` command described next should be used.

6.13 Compiling a complete system from scratch

Under exceptional circumstances it might be useful to compile a complete system's source from scratch, e.g. directly after restoring a complete system version by the `epx_restore` command described above.

The command syntax is as follows:

```
epx_make
```

and may be executed only by the local site administrator. This command completely re-builds the EUROPLEXUS system starting from the current sources:

- it compiles all Fortran sources and updates the module files (`.mod`) and the object library (`libplex.lib`);

- it builds and updates the standard executable `europlexus.exe`;

- it executes all the benchmark tests;

- it compiles and updates the user's manual in all its versions.

6.14 Finding the correct compilation order

A problem related with the use of F90 modules is that, as already mentioned in Section 6.4, the order in which code source files are compiled does matter, unlike with the previous Fortran 77 standard.

Therefore, a command procedure has been prepared that is able to search the correct compilation order for any number of source files, i.e. ranging from a few files under modification, to the full source of the code.

The command syntax is as follows:

```
epx_ordo
```

This command scans the current directory for any source files (`*.ff`), then searches module dependencies and module usage and finally produces a list of the source files in the correct compilation order in a file named `epx_ordo.txt` in the current directory. This file may then be used by other commands that use it implicitly, such as for example the `epx_test_sources` procedure (see Section 6.4).

The command is based upon a Perl procedure (`epx_ordo.pl`) and an associated Fortran program (`ordo.f`). These files have been slightly adapted to the Windows operating system starting from the equivalent ones that had been developed by H. Bung for the Central mirror site (under Unix).

Searching the compilation order for the entire EUROPLEXUS source takes less than 10 seconds of CPU on a state-of-the-art PC, so this command may be used as often as necessary.

6.15 Preparing for a "batch" evolution

In April 2002 a new possibility has been added at the central.mirror site, that of submitting a set of files for evolution as a single, tarred gzipped package. This may be useful to speed up evolution request operations under various circumstances and especially for evolving large numbers of files.

To use this feature, users must lock files as usual and perform all the normal check operations at their mirror site. When the files to be evolved are ready, a tar file containing all the sources has to be prepared by a command such as `tar cvf evo.tar *.*`, compressed by `gzip -9 evo.tar`, and finally renamed: `mv evo.tar.gz evo.tgz` (note in fact that the final suffix must be `.tgz`).

The contents of this file mjust be as follows:

- Any number of source files `*.ff`;
- Any number of manual files `*.ttx`;
- Any number of benchmark files `*.epx`. For each of these, if a corresponding mesh file `.msh` exists in the EUROPLEXUS bench directory, then it must also be included in the package (even though it has not been modified). A matching `.zip` file may be optionally included in the package;
- For each of the above source files `*.ff`, `*.ttx` or `*.epx`, a corresponding text file `*.txt`, containing the evolution comments must also be present in the package. These comment files must not be empty (there must be at least one line of comments);
- The package should not contain any other extraneous file besides the ones listed above;
- All file names must be in lower case.

Since ensuring all the above requirements may be tedious, a command has been prepared that performs automatically a series of checks. Assume that a user has prepared a series of files for evolution. Comments for these files have been stored in a single file `evo.txt` wity the following syntax: one line containing just the file name, followed by one or more non-blank lines containing the comments, followed by one or more blank lines, and so on for each file to be evolved. Then the command:

```
evototgz
```

will perform the checks listed above and produce the package `evo.tgz`, ready for evolution, in the current (work) directory. Be aware that the command may not verify the proper locking of the files, so this operation has still to be done manually.

In July 2009, a new procedure epx_make_evo has been prepared which facilitates the creation of the `evo.txt` file containing the list of the modified source files to be evolved. In addition, a file `lock.txt` is created, which can be used to lock all the files in the central mirror site.

The files to be evolved are automatically found on the current directory as being the valid source file names (`*.ff`, `*.epx`, `*.ttx`) which have been modified with respect to their current standard version.

The command syntax is as follows:

```
epx_make_evo [-f fiche_number] [-s] [-l]
```

where:

`-f`	Introduces the number of the fiche de developpement associated with the evolution.
`-s`	The evolution has no associated fiche.

`-l` Creates only the lock file.

The `-f` and `-s` switches are mutually exclusive. If none is giver, fiche number 0 is assumed.

After producing an evolution package with `evototgz`, it may be tested locally (if it contains any source files) by the command:

```
epx_test_evo
```

This procedure unpacks the evolution package in a special sub-directory, checks the file stamp by the `epx_newer` command, then it retrieves all dependencies, it compiles and links the code and finally it passes all benchmarks. It takes no action on benchmarks, manuals and validation files which might be present as well in the evolution package.

6.16 Updating the file header

In September 2004 a new command has been added, which may be used to reset the "header" of a source file (or various files) to the value it has in the current standard version of the file itself.

The command syntax is as follows:

```
epx_update_header [-i] [-b] [-m] [-v] file(s)[.ext]
```

where:

`-i` The file to be treated is an include file, not a source file.

`-b` The file to be treated is a bench file, not a source file.

`-m` The file to be treated is a manual file, not a source file.

`-v` The file to be treated is a validation file, not a source file.

`name` Name of the file, with or without extension.

7 Organisation of the User's manual

The EUROPLEXUS User's manual describes the syntax of all available input directives. The document is continuously updated together with the code itself, and is part of the EUROPLEXUS distribution, see Section 4.2.

The manual language is English. Versions in other languages may be set up and maintained locally, if desired.

7.1 Document preparation language

The LaTeX document preparation system [7], based upon D. Knuth's TeX [8], has been chosen for the manual. This has the drawbacks, compared with more recent document management systems or word processors, of not being interactive, and of rendering the inclusion of graphics rather cumbersome. The former is actually a drawback only for the preparation of manual developments, but it may even be viewed as a plus during an automatic evolution. The latter is not a serious shortcoming for a User's manual.

On the positive side, LaTeX is freely available on the network for a variety of platforms. The source files are pure text (ASCII) files and may therefore be transferred without problems across sites and on different platforms. The treatment of mathematical formulas is excellent.

Recently, a new version of LaTeX known as LaTeX2e has appeared [9]. The manual files themselves do not use any feature exclusive to the new version, and so could be compiled with either the old or the

new LaTeX. However, some of the extra LaTeX packages that are loaded in the manual main input file, e.g. to produce customized headers and footers, to get hyperlinks in PDF output, etc, do require the new version of the LaTeX compiler. See details on this below in Section 7.5.1 and 7.5.9.

7.2 Hierarchical structure

The manual is subdivided into Sections, and each of these into logical "pages". Each logical page contains the description of a well defined input directive, and may consist of one or more (typically not more than 2 or 3) physical pages of the manual.

Each Section has an associated acronym or letter, e.g. INT for the Introduction, A for Problem Type and Dimensioning, B for Mesh and Grid Motion, C for Materials etc. Each logical page has a unique index consisting of the Section acronym followed by a period and by a number. For example, logical page C.110 describes the linear elastic material, C.150 the concrete material, etc. These numbers appear in growing order in the manual and are usually spaced of 10 from one another in order to allow insertion of new logical pages, if necessary.

The logical page index appears at the center of each physical page header, and is used for internal cross-referencing. For example: "see linear elastic material on Page C.110" means the logical page and has nothing to do with the physical page number. The right part of the page header contains the date (month and year) of the last modification that affected the logical page. The central part of the page footer contains the physical page number.

The manual source is organised hierarchically. A LaTeX main file (`gbnotice.ttx`) contains the main document settings and a series of `\input{file.tex}` statements, one for each Section of the manual. This instructs the LaTeX compiler to read in the mentioned file `file.tex`. Each Section file contains in turn just a series of `\input` statements, one for each logical page.

Note that the suffix `.ttx` applies to unfiltered manual source files, while the suffix `.tex` denotes the filtered files. For more information on filtering issues see Sections 5.1 and 5.10.

Logical page files have names of the form `gbSSS_NNNN.ttx` where `SSS` is the Section acronym (`INT` or `A` or `B` etc.), while `NNNN` is the logical page number, with leading zeroes as needed. For example, the name of the file containing the logical page C.110 is `gbc_0110.ttx`.

This relatively fine subdivision is useful in the present multi-site development environment in order to reduce the probability that, when a developer wants to evolve a certain directive description, he or she finds the corresponding file locked by another developer.

7.3 Manual target formats

The following target formats of the Users' manual are automatically produced, starting from a unique LaTeX source:

- LaTeX device independent (`.dvi`) format: this may be visualized and printed, for example by the `yap` previewer contained in the MiKTeX package, see Section 7.4);
- Adobe PostScript format: this is mostly suitable for printing, but may be visualized on screen for example by Ghostscript/GSview (see Section 7.4);
- Adobe Portable Document Format (PDF): this is suitable for on-line distribution and may be visualized and printed on any platform by the Acrobat Reader sofware from Adobe, freely available for a variety of platforms (http://www.adobe.com/products/acrobat/readstep.html);
- HyperText Markup Language (HTML) format: this is suitable for browsing on the network. Since the standard is evolving, two different HTML versions of the manual are produced, each having strengths and weaknesses, see Section 7.4 for more details.

7.4 Tools under Windows

The chosen version of the LaTeX software, among the several ones available on the network for the Windows environment, is the one known as MiKTeX (http://www.miktex.org), maintained by Christian Schenck. According to the MiKTeX home page, the distribution contains (the list below refers to the version of end 2000):

- TeX 3.14159 [The traditional TeX compiler];
- Yap (Yet Another Previewer) 0.97 [A tool to view TeX output];
- METAFONT 2.718 [Font compiler];
- LaTeX2e <99/06/01> [Macro package];
- Standard LaTeX packages [Babel, psnfss, etc.];
- Dvips 5.86 [Converts TeX output into PostScript];
- Dvipdfm 0.12.6e [Converts TeX output into PDF];
- pdfTeX 0.14d [A variant of the TeX compiler which produces PDF];
- Computer Modern PostScript Fonts [Standard TeX fonts in Type1 format];
- MakeIndex 2.12 [A tool to produce indexes];
- etc.

The MiKTeX distribution also contains a utility called `tth` (developed by Ian Hutchinson, http://hutchinson.belmont.ma.us/tth/) that can transform a TeX or LaTeX file into an HTML file. This is the free version of the utility, which has somewhat limited functionality: it produces a single HTML output file, which is therefore rather large and slow to load on remote browsers. A commercial version of `tth` is also available, which may split the file into smaller pieces. More on this below in Section 7.5.4.

The MiKTeX package, however, does not contain a PostScript interpreter and on-screen visualizer. For this task, the Ghostscript/GSview software has been obtained from http://www.cs.wisc.edu/~ghost/. This allows to preview on screen the PostScript version of the manual before printing it, to print selected page ranges, etc.

Another very useful tool, which may be used to convert a PostScript file into an (editable) vector format, such as for example Windows Meta File (WMF), which may be imported and edited in Microsoft Word, is pstoedit, by Wolfgang Glunz. This package may either be used from the command line, or be installed as a plug-in of Ghostscript/GSview (see above). In the latter case, a PostScript file may be visualized in GSview and then saved for example in WMF from GSview itself. The pstoedit package may be found under: http://www.pstoedit.net/pstoedit (but is now included in the GSview distribution).

Another tool that is used in the preparation of the manual is the `hevea/hacha` utility that, similarly to `tth`, transforms a LaTeX file into HTML. The advantage over `tth` is that by means of `hacha` the HTML file may be split in a series of smaller files, that load much more quickly on remote browsers. These two utilities are developed at INRIA by Luc Maranget (http://para.inria.fr/~maranget/hevea/) in the Objective Caml language. A port to Windows has been prepared by Philip A. Viton of the Ohio State University (http://www.arch.ohio-state.edu/crp/faculty/pviton/support/).

7.5 LaTeX compilation passes

The production of the manual in the different target formats is driven by the perl script `epx_test_manuals`, as already mentioned in Section 6.7. This script proceeds as follows:

- First, it retrieves (i.e., copies to the current directory) from the EUROPLEXUS `manual` directory the manual template file `gbnotice.ttx`, if this files is not already present locally. The other LaTeX manual input files do not need to be present locally, because the MiKTeX input search path is extended so as to contain the EUROPLEXUS `manual_filtered` directory, so that any `.tex` files not present locally are found there (see Section 7.5.10).

- Next, it builds the various manual formats, except the PDF format, by calling the following four lower-level scripts: `epx_pass_1`, `epx_pass_2`, `epx_pass_3` and `epx_pass_4`;

- Finally, it builds the PDF format by calling the following three lower-level scripts: `epx_pass_1_pdf`, `epx_pass_2_pdf`, and `epx_pass_3_pdf`.

The main reason for having several low level scripts, each of which performs a LaTeX (or other) compilation ("pass"), is that a LaTeX source must be compiled repeatedly in order to get the correct cross-references, table of contents, index etc.

Each script prepares an ad-hoc version of the main LaTeX input file by appropriate filtering (via the `epx_filter_manual` command, described in Section 5.10.1) of the template file `gbnotice.ttx`, that is listed in Section 7.5.1. The filtering process applies the modifications appropriate for the current compilation pass, and saves the modified file locally with the name `gbnotice.tex`. (a file with the same name is overwritten if already present).

The template input file and the different Users' manual compilation passes are shortly described below.

7.5.1 LaTeX template input file

The template LaTeX input file `gbnotice.ttx`, which is also listed in Appendix, has the following structure:

```
\documentclass[11pt]{article}
%
\usepackage{fancyhdr}
%====================================================================
CIF HEVEA
\usepackage{makeidx}
CENDIF
%====================================================================
\usepackage{time}
%
\newcommand{\unit}[1]{{\rm\,#1}}                    % Ex.: 25\unit{km}
\newcommand{\gradi}{\ifmmode^\circ\else$^\circ$\fi} % Ex.: 90\gradi
\newcommand{\celsius}{\ifmmode{}^\circ{\rm C}%
\else${}^\circ{\rm C}$\fi}                          % Ex.: 25\unit{\celsius}
\newcommand{\newpar}{\par\vspace{0.5cm}}
%
\setlength{\headheight}{0.5cm}
\setlength{\headsep}{0.5cm}
\setlength{\oddsidemargin}{0.0cm}
\setlength{\topmargin}{0.0cm}
\setlength{\marginparwidth}{0.cm}
\setlength{\marginparsep}{0.cm}
\setlength{\textwidth}{16.0 cm}
\setlength{\textheight}{22.0 cm}
%
\makeindex
%====================================================================
CIF PDFLATEX
\usepackage[pdftex,a4paper=true,colorlinks=true]{hyperref}
CENDIF
%====================================================================
%====================================================================
CIF HEVEA
%\renewcommand{\cuttingunit}{subsubsection}
\setcounter{cuttingdepth}{3}
CENDIF
```

```
%=====================================================================
\begin{document}
%=====================================================================
CIF PDFLATEX
%   This forces the PDF page to be A4 (it seems that the above
%   a4paper switch in the pdftex package usage has no effect!)
\pdfpagewidth=21cm
\pdfpageheight=29.7cm
CENDIF
%=====================================================================
\input{gbtit.tex}
%
\newpage
\lhead{\bf EUROPLEXUS}
\chead{}
\rhead{\bf Contents}
\lfoot{}
\cfoot{\today\ at \now}
\rfoot{\thepage}
%
\renewcommand{\headrulewidth}{0.4pt}
\renewcommand{\footrulewidth}{0.4pt}
\pagestyle{fancy}
\thispagestyle{plain}
\tableofcontents
%
\newpage
%
\input{gbint.tex}
\input{gba.tex}
\input{gbb.tex}
\input{gbc1.tex}
\input{gbc2.tex}
\input{gbc3.tex}
\input{gbc4.tex}
\input{gbc5.tex}
\input{gbd.tex}
\input{gbe.tex}
\input{gbf.tex}
\input{gbg.tex}
\input{gbh.tex}
\input{gbi.tex}
\input{gbed.tex}
\input{gbsr.tex}
\input{gbrm.tex}
\input{gbex.tex}
\input{gbbib.tex}
%=====================================================================
CIFNOT HEVEA
\input{gbindex.tex}
CENDIF
%=====================================================================
%=====================================================================
CIFNOT LATEX1 HEVEA
%   On first LaTeX compilation (and under Hevea compilation),
%   comment out \input{gbnotice.ind}.
%   Then, type "makeindex gbnotice.idx" to update gbnotice.ind.
%   Then uncomment and recompile 2 times to get correct doc with index.
\input{gbnotice.ind}
CENDIF
%=====================================================================
%
%=====================================================================
CIF HEVEA
\printindex
CENDIF
%=====================================================================
%
\end{document}
```

Lines starting with % are treated as comments.

The first active line, containing the \documentclass directive, is a LaTeX2e command that sets the document style to be article, and the normal point size to be 11 points.

The next line \usepackage{fancyhdr} loads an accessory LaTeX macro package that is used to produce customized headers and footers. This package requires the LaTeX2e version of the compiler. A few such non-standard LaTeX packages are used to obtain special effects, as summarized in Section 7.5.9.

The next line \usepackage{makeidx} is contained within a CIF HEVEA … CENDIF filtering conditional and therefore it is activated only when the file is to be processed by hevea instead of the LaTeX compiler (to obtain this behaviour, the filter must be invoked by adding the HEVEA keyword to the standard, platform-related keywords).

The next line \usepackage{time} loads a package needed to get the current time.

The following lines starting with \newcommand introduce some useful new commands.

Next come several settings of the document basic dimensions (width, height, margins etc.), all starting with the \setlength directive.

The \makeindex line instructs the LaTeX compiler to collect the information for producing an index.

The conditional line \usepackage[pdftex,a4paper=true]{hyperref} loads the hyperref package, which used in conjunction with the pdflatex command produces a PDF version containing hyperlinks in place of the normal LaTeX cross-references.

The next conditional line \setcounter{cuttingdepth}{3} sets the "cutting depth" by which hevea/hacha splits the HTML file into pieces. This terminates the document preamble.

The rest of the file, contained between \begin{document} and \end{document}, is the document body. It consists mainly of \input directives, as already noted. Directives of the form \lhead{}, \chead{}, \lfoot{} etc. are processed by the fancyhdr package, in order to set the page headers and footers.

The \tableofcontents directive produces the table of contents before the manual main body.

Finally the two conditional directives \input{gbnotice.ind} and \printindex are used to produce the index at the end of the document.

7.5.2 First pass

The first pass of manual compilation, performed by script epx_pass_1.pl, cleans up any residual manual files from a possible previous compilation of the manual (by calling the script epx_clean_manual_files.pl), filters the template file by the standard local keys plus the keyword LATEX1 (this removes the line \input{gbnotice.ind}) and runs the LaTeX compiler.

All cross references remain unresolved, but a file gbnotice.aux is produced that contains cross-reference information for the next pass.

A file containing index information, gbnotice.idx, is also produced.

Then, the script runs the makeindex utility, which is part of the MiKTeX distribution and which processes the raw index information contained in file gbnotice.idx, to produce an index file gbnotice.ind, suitable for being included in the document (see next step).

7.5.3 Second pass

The second pass of manual compilation, performed by script `epx_pass_2.pl`, filters the template file by the standard local keys (the line `\input{gbnotice.ind}` remains therefore active), and then runs the LaTeX compiler for the second time.

This time cross references should be right, but the index, which is produced by the `\input{gbnotice.ind}` command, could be wrong because based on obsolete (first-pass) page numbering information.

The `makeindex` utility is therefore run again to produce a correct index file `gbnotice.ind`, for the next step.

7.5.4 Third pass

The third pass of manual compilation, performed by script `epx_pass_3.pl`, filters the template file by the standard local keys (the line `\input{gbnotice.ind}` remains therefore active), then runs the LaTeX compiler for the third time.

This time both the cross references and the index should be right, so the resulting `.dvi` file is retained as the final one and renamed `manual.dvi`.

Next, the script processes the `dvi` file by means of `dvips`, which produces the PostScript version of the document, file `manual.ps`.

Finally, the script processes the LaTeX source file (`gbnotice.tex`) through the `tth` filter (instead of the LaTeX compiler), to produce the HTML version of the document, `manual.html`.

For unknown reasons, the index which appears in `manual.html` contains page numbers (which are useless in an HTML version) instead of the active links that would be expected. However, the table of contents and the internal cross references do appear as active links. The problem with the index may be due to a bug in `tth`, or to an error in the paramaters passed to the utility. Investigation is underway but the problem is not solved yet.

Another yet unsolved problem with `tth` is that the filter does not seem to recognize the input switch (that according to the manual should be `-p`) which tells it to search input files from an additional directory (the standard EUROPLEXUS manual directory) whene these are missing on the current directory. The problem is temporarily bypassed by the following procedure:

- a temporary sub-directory `tth` is created;
- all `*.tex` files from the standard EUROPLEXUS filtered manual directory are copied to it;
- all `*.tex` files from the evolution directory are copied to `tth` (thus overwriting the homonyms from the EUROPLEXUS manual directory);
- all auxiliary files `gbnotice.*` from the evolution directory are copied to `tth`;
- the procedure changes directory to `tth` and invokes `tth`;
- finally the procedure changes back to the evolution directory, copies the `tth` resulting files from the `tth` directory and deletes the `tth` directory with all its contents.

7.5.5 Fourth pass

The fourth pass of manual compilation serves to produce another HTML version of the manual by means of the `hevea/hacha` utility. Actually, two versions are produced: a “monolithic” HTML file `manual_h.html`, similar to the one produced by `tth` in pass 3, and a “split” file (actually a series of

smaller HTML files contained in a separate subdirectory `hacha`), that is produced by applying `hacha` to the monolithic file.

The entry point of the split document is `hacha\index.html`, which contains only the table of contents. By following the links, one can open a Section file (numbered `manual_h001.html` to `manual_h018.html`) or the index file (which happens to be `manual_h019.html`).

All hyperlinks work, both in the table of contents and in the index, which is a plus with respect to the `tth` version. However, the formatting of some manual parts obtained with `tth` is nicer.

To generate these versions, the `epx_pass_4.pl` script filters the template file by the standard local keys plus the keyword `HEVEA`. This produces a series of changes:

- Activate the `\usepackage{makeidx}` line;
- Activate the `\setcounter{cuttingdepth}` line;
- Deactivate the `\input{gbindex.tex}` line;
- Deactivate the `\input{gbnotice.ind}` line
- Activate the `\printindex` line.

Then the script runs the `hevea` compiler, and finally the `hacha` utility.

7.5.6 First PDF pass

The manual processing continues with the production of the PDF version. This takes place in three passes, similar to the first three passes seen above in Sections 7.5.2 to 7.5.4. The first pass of manual compilation, performed by script `epx_pass_1_pdf.pl`, cleans up any residual manual files from previous compilation of the manual, filters the template file by the standard local keys plus the keywords `LATEX1` and `PDFLATEX`, and runs a special version of the LaTeX compiler (`pdflatex`) that produces PDF output directly, instead of the normal DVI output. The filtering process eliminates the line `\input{gbnotice.ind}` like in the first pass of LaTeX compilation (keyword `LATEX1`), and it activates the lines starting with `\usepackage[pdftex`, `\pdfpagewidth` and `\pdfpageheight` (keyword `PDFLATEX`).

Then, the script runs the `makeindex` utility.

7.5.7 Second PDF pass

The second PDF pass, script `epx_pass_2_pdf.pl`, is also quite similar to the one described in Section 7.5.3. The only difference is that, like in the first PDF pass, the special keyword `PDFLATEX` is used in the filtering process, and that of course the `pdflatex` command is used instead of the `latex` command. The `makeindex` utility is run again at the end of the script.

7.5.8 Third PDF pass

The third and final PDF pass, script `epx_pass_3_pdf.pl`, is quite similar to the second one described in Section 7.5.7. The same filtering is performed, and the PDF file obtained from `pdflatex` is considered as the final version and renamed `manual.pdf`.

7.5.9 Summary of non-standard LaTeX packages

The non-standard LaTeX packages used in the production of the manual are summarized in Table 7. They may be freely obtained from the Comprehensive TeX Archive Network (CTAN) at the address: http://tug2.cs.umb.edu/ctan/).

Note that the `makeidx` package (for index preparation) and `hyperref` package (for hyperlink generation in the PDF version), which are also used, are standard packages, contained in the MiKTeX distribution.

Table 7 - Summary of non-standard LaTeX packages used for the manual

Purpose	Name	Type/Location
Custom headers/footers	`fancyhdr.sty`[1]	http://tug2.cs.umb.edu/ctan/
Current time	`time.sty`	http://tug2.cs.umb.edu/ctan/
LaTeX to HTML translation	`hevea.sty`	Contained in the `hevea` package
Comments (used by `hevea.sty`)	`comment.sty`	http://tug2.cs.umb.edu/ctan/

To install these packages on a standard MiKTeX installation perform the following steps:

- Put the new style files in a separate directory in the MiKTeX installation. For example, assuming that MiKTeX is installed on `C:\Texmf`, the directory (to be created anew) could be `C:\Texmf\Tex\Latex\Folco`;

- Log-in as administrator, then from a console window type the following command, to update the MiKTeX file name database:

```
initexmf --update-fndb
```

- Log-in again as normal user and verify that LaTeX finds the new file names (for example, `fancyhdr.sty`) on the updated data-base, by issuing the command:

```
initexmf --find-latex-input comment.sty
```

7.5.10 Extending the MiKTeX input search path

It is useful to extend the standard MiKTeX search path for input files, so that filtered input files (`.tex`) that are to be loaded via an `\input{file.tex}` directive are taken by default from the EUROPLEXUS `manual_filtered` directory, if they are not present in the current directory.

This may be obtained in two ways: by adding a personal configuration file or by modifying the global configuration file of the MiKTeX system. The first method is as follows:

- A Personal Configuration File for MiKTeX (`.ini`) is prepared. Let the name of this file be, say, `%EUROPLEXUS%\util\miktex_folco.ini`.

- This file is best produced by copying the MiKTeX Global Configuration File `miktex.ini` from subdirectory `miktex\config` of the MiKTeX installation directory (`C:\Appl\Texmf` at JRC).

- Then, edit this file and remove all "sections" except those labeled LaTeX, pdfLaTeX, pdfTeX and TeX.

- Each of the remaining sections contains an entry `Input Dirs=...`, which specifies where the corresponding program (LaTeX etc.) searches for input files. Add to the list of directories the following string:

[1] Note that this package was non-standard only in previous versions of MiKTeX. It is contained by default in the "large" installation of version 2.1.

```
;E:\EUROPLEXUS\manual_filtered
```

- Finally, update the MiKTeX configuration by the command:

```
initexmf --personal=E:\EUROPLEXUS\util\miktex_folco.ini
```

- Note that this activates the new search paths for the current user only. Verify that the new search path is active by a command such as:

```
initexmf --find-latex-input gba_0010.tex
```

The file `miktex_folco.ini` should be as follows after editing (it is also listed in Appendix):

```
;; -------------------------------------------------------------------------
;; LaTeX
;; -------------------------------------------------------------------------

[LaTeX]

;; Where LaTeX searches for input files.
Input
Dirs=.;%R\tex\latex//;%R\tex\generic//;%R\tex//;E:\EUROPLEXUS\manual_filtered

;; Common LaTeX filename extensions.
Extensions=.tex;.ltx

;; -------------------------------------------------------------------------
;; pdfLaTeX
;; -------------------------------------------------------------------------

[pdfLaTeX]

;; Where pdfLaTeX searches for input files.
Input
Dirs=.;%R\pdftex\latex//;%R\pdftex\generic//;%R\pdftex//;%R\tex\latex//;%R\tex\g
eneric//;%R\tex//;E:\EUROPLEXUS\manual_filtered

;; -------------------------------------------------------------------------
;; pdfTeX
;; -------------------------------------------------------------------------

[pdfTeX]

;; Where pdfTeX searches for input files.
Input
Dirs=.;%R\pdftex\plain//;%R\pdftex\generic//;%R\pdftex//;%R\tex\plain//;%R\tex\g
eneric//;%R\tex//;E:\EUROPLEXUS\manual_filtered

;; Where pdfTeX searches for PostScript headers.
PSPath=.;%R\pdftex//

;; -------------------------------------------------------------------------
;; TeX
;; -------------------------------------------------------------------------

[TeX]

;; Where TeX looks for input files.
Input
Dirs=.;%R\tex\plain//;%R\tex\generic//;%R\tex//;E:\EUROPLEXUS\manual_filtered

;; Common TeX filename extensions.
Extensions=.tex
```

Note that some very long lines above appear folded. They must not be folded in the configuration file.

This method has the advantage that it does not require any change to the MiKTeX system, which is useful e.g. in case of frequent upgrades of the system itself. The drawback is that, as indicated, the new search path is activated for the current user only. This has as an effect that the manual update procedure

will execute correctly when explicitly executed by the site administrator, but will fail when started up automatically by the `at` or `winat` command. In fact, see Section 6.3.1, the `at` command executes the procedure under the SYSTEM user, not as the site administrator, and unfortunately there seems to be no way to log in onto a Windows machine as the SYSTEM user, and hence it is impossible to set the correct LaTeX path via a personal configuration file for this user.

A possible solution to this problem is the second method mentioned above. This consists in modifying the Global Configuration File (at JRC this is file `miktex.ini` from subdirectory `miktex\config` of the MiKTeX installation directory) by adding to the four sections labeled LaTeX, pdfLaTeX, pdfTeX and TeX the extra directory path listed above.

This makes it possible to execute the procedure via `at`, but has the drawback that the changes must be re-applied every time the MiKTeX system is updated.

Another configuration command that may be useful is the command that removes the reading of a personal configuration file (if previously set), which is:

```
initexmf --personal
```

Finally, note that activation or deactivation of the personal configuration file, and any changes in it or in the global configuration file are immediately effective: they do not require re-initializing the MiKTeX system.

7.6 Converting accented and other special characters

A problem that may arise in the development of the User's Manual or of other documents via the LaTeX package is the treatment of accented and other special characters. LaTeX does not recognize these characters directly (unless a special style package, e.g. `t1enc.sty`, is used), but uses its own special escape sequences. For example, the "a" letter with a grave accent is indicated by `\`{a}`.

In order to facilitate the port to LaTeX of text containing such special characters, a filter has been prepared `epx_accents`. The filter converts to the appropriate LaTeX sequences nearly all characters between positions 192 to 255 of the ASCII extended table (a few very special reserved characters are not translated). These characters include all accented letters.

To invoke the filter, type the command:

```
epx_accents name(s)[.ttx]
```

where:

`name(s)` Name of the file to be filtered, with or without extension. If present, the extension must be `.ttx`. Multiple files may be specified by using wildcard characters. Each file is directly converted (modified), so beware that the old version is lost.

The filter text (a Perl procedure) is listed in the Appendix.

7.7 Testing a single manual source file

When developing the manual, modifications on local manual files may be thoroughly tested by the command `epx_test_manuals`, that has been described in Section 7.5. However, this produces a complete set of manuals in the various formats, and is a somewhat lengthy operation.

It is sometimes useful to test just one manual file, because it is both faster and simpler to locate possible LaTeX errors.

A command for performing this operation has been prepared, `epx_test_man`. The syntax is as follows:

```
epx_test_man name[.ttx]
```

where:

`name` Name of the file to be tested locally, with or without the extension `.ttx`.

Only the `.dvi` version of the result file is produced, and it is always named `driver.dvi`.

Note that since the file is compiled separately by LaTeX, features (such as for example cross references and section numbering) that depend upon other files in the manual will *not* be treated correctly. But the command is very useful to rapidly eliminate nasty syntax errors in the LaTeX file. Remember, however, that a final check of the file (or files) by the more complete `epx_test_manuals` command is mandatory, e.g. before submitting the modified manual input files for evolution.

8 Interactive development

This Section describes the interactive development of EUROPLEXUS by means of the MicroSoft Developer Studio software. As anticipated in Section 2.2, the Developer Studio GUI-based compilation environment offers a debugging facility, while this is not available for the command-line driven environment. This property renders the Developer Studio tool preferable for interactive code development.

In Section 8.1 below, the set-up from scratch of a Developer Studio "Workspace" suitable for the development of EUROPLEXUS is described in some detail. This is not a simple operation and requires many detailed settings. However, once set up the environment, the development and debugging of a test version of the code becomes quite efficient. For example, module dependencies are automatically treated, which is not the case in the command-line environment.

To facilitate everyday work of ordinary code developers, a perl script has been prepared which automatically produces, on the local current directory, a new Developer Studio project suitable for the EUROPLEXUS development. This is described in Section 8.3.

8.1 Preparing an EUROPLEXUS workspace from scratch

In order to develop EUROPLEXUS under Microsoft Developer Studio, set up a new Workspace (say `plex`) containing two Projects: `devel` and `eplex`. The `devel` project contains the unfiltered "`*.ff`" Fortran source files, while the `eplex` project contains the corresponding filtered sources "`*.f`", which may actually be compiled. The filtering process is automatically performed since it is set up as a "custom build step" action. Hereafter, we describe the set up of a Workspace of type Console application, which is the standard for EUROPLEXUS (since May 2003). The set up of a QuickWin application would be slightly different, and the variations are marked like this in parentheses when appropriate.

To set up the Workspace, do the following:

1. Open Microsoft Developer Studio (***Start → Programs → Visual Fortran → Developer Studio***) and click on ***File → New → Workspace***. Click the ***Workspaces*** tab and select ***Blank Workspace***.

   Under ***Workspace name*** type in `plex`, and under ***Location*** type your directory of choice (e.g.: `E:\EUROPLEXUS\Development`); this creates a workspace file `E:\EUROPLEXUS\Development\plex.dsw` and a couple of associated files (`plex.ncb` and `plex.opt`);

2. A Developer Studio window appears, that contains "Workspace plex: 0 projects". Click on this sentence once to highlight it, then press the right mouse button and click on ***Add new project to workspace***.

   The ***New*** dialog appears, choose the ***Projects*** tab if necessary and choose as project type ***Fortran Console Application*** (***Fortran Standard Graphics or QuickWin Application*** in the QuickWin case). As project name type in `eplex`, and as ***Location*** choose e.g.

`E:\EUROPLEXUS\Development`. Activate the ***Add to current workspace*** radio button and make sure that the ***Win32*** box is checked in the ***Platforms*** field. Then click on ***OK***.

A dialog ***Fortran Console Application - Step 1 of 1*** appears, activate the ***An empty project*** radio button (QuickWin case: ***Fortran Standard Graphics or QuickWin Application - Step 1 of 1*** appears, activate the ***QuickWin (multiple windows)*** radio button). Click on ***Finish***, then on ***OK***.

The Developer Studio window should now contain "Workspace plex: 1 projects", and a project entry "eplex files", with three sub-entries (empty for the moment): Source Files, Header Files and Resource Files. The `E:\EUROPLEXUS\Development` directory now contains, in addition to the previous three files (`plex.dsw`, `plex.ncb` and `plex.opt`), the project file `eplex.dsp` and a subdirectory `debug` (for the "Debug" version of the eplex project).

Note that when a new project is created in Developer Studio, by default two "Configurations" are foreseen: Debug and Release. New configurations may be added as needed, or an existing one (including either Debug or Release) may be deleted. The currently active configuration is set by ***Build → Set Active Configuration***.

Because the main purpose of the `eplex` project is interactive version development and debugging, the Debug configuration should be constantly activated. The Release configuration may either be ignored or be removed from the project. The same applies also to the devel project to be added below. To remove a configuration go under ***Build → Configurations***, highlight the chosen configuration, click on Remove and then confirm your choice.

3. Now add a second project `devel` to the `plex` Workspace. Click on the "Workspace plex: 1 projects" to highlight it, then press the right mouse button and click on ***Add new project to workspace***.

   The ***New*** dialog appears, choose the ***Projects*** tab if necessary and choose as project type ***Win32 Static Library***. As project name type in `devel`, and as ***Location*** choose e.g. `E:\EUROPLEXUS\Development`. Activate the ***Add to current workspace*** radio button and activate the ***Dependency of*** box, by choosing `eplex`. Make sure that the ***Win32*** box is checked in the ***Platforms*** field. Then click on ***OK***.

   A dialog ***Win32 Static Library - Step 1 of 1*** appears: leave the two boxes empty. Click on ***Finish***, then on ***OK***.

   The Developer Studio window should now contain "Workspace plex: 2 projects", and two project entries: "devel files" with two sub-entries Source Files and Header Files, and "eplex files", with four sub-entries: Source Files, Header Files, Resource Files and devel. The `E:\EUROPLEXUS\Development` directory now contains a new project file `devel.dsp`, and a new sub-directory `devel__Win32_Debug` (for the Debug configuration of the devel project). You may remove this sub-directory.

4. Click on ***File → Save All*** to save the work done so far (do this from time to time).

5. Next, let us add some source files to the projects. It is always advisable that the main program file (`main.ff` in EUROPLEXUS) be included in any Developer Studio project, so we will show how to add this file as an example. Any other source files may be added in the same manner (QuickWin case: add also at least `ifwin.ff` and `m_qwin.ff`).

   In a console window located in directory `E:\EUROPLEXUS\Development`, type the command `epx_get main`, which retrieves the source file `main.ff` from the EUROPLEXUS sources library.

   Then, activate the devel project if necessary (***Project → Set Active Project***, then choose devel), click in the Source Files entry of devel files, press the right mouse button and click on ***Add files to folder***.

A dialog ***Insert Files into Project*** appears: in the ***Look in*** space select the Development directory in `E:\EUROPLEXUS`; in the ***File name*** field type `main.ff` (or select ***All files*** from the ***Files of type*** drop-down menu, then click on `main.ff`) and click on ***OK***. The file `main.ff` is added to the devel project Source Files entry.

6. Next, we set the Custom Build step for the added file `main.ff`, in order to filter it and transform it into `main.f`. In the Workspace graphical tree click on ***devel files → Source Files***, press the right mouse button , click on ***Settings*** and select the ***Custom Build*** tag. Under ***Commands***, type:

```
E:\EUROPLEXUS\Util\epx_filbat.bat $(InputDir)\$(InputName)
```

In the QuickWin case, instead:

```
E:\EUROPLEXUS\Util\epx_filbat_quickwin.bat $(InputDir)\$(InputName)
```

The `epx_filbat.bat` file contains the following simple commands:

```
@echo off
echo Filtering %1.ff
\\SM48\E\EUROPLEXUS\Util\epx_filter.exe WIN OGL <%1.ff >%1.f
echo Filtering done
```

The `epx_filbat_quickwin.bat` file is similar, except that it uses the additional compilation keyword `QWIN`.

Under ***Outputs*** type:

```
$(InputDir)\$(InputName).f
```

This has the effect of invoking the `epx_filter` utility described in Section 5.1 with the two (hard-coded) passwords `WIN OGL` on file `$(InputDir)\$(InputName).ff` and placing the filtered output in file `$(InputDir)\$(InputName).f`. Here `$(InputDir)` is `E:\EUROPLEXUS\Development` and `$(InputName)` is for example `main`.

Having selected the whole Source Files entry rather than a specific file (`main.ff`) when applying the Custom Build setting, has the effect that the custom build action is associated to all files currently present in this entry (but unfortunately not to those added in the future).

7. To test the applied setting, click on `main.ff` to highlight it, then click on ***Build → Compile main.ff*** (or press `CTRL+F7`). In the Build pane of the Developer Studio window should appear the message:

```
Performing Custom Build Step on ..\main.ff
Filtering \EUROPLEXUS\Development\main.ff
1
Filtering done
```

and a file `main.f` should be created in directory `E:\EUROPLEXUS\Development`.

8. Next, we add the filtered file `main.f` to the `eplex` project. Click on the Source Files entry of the eplex files entry, press the right mouse button and click on ***Add files to Folder***.

A dialog ***Insert Files into Project*** appears: in the ***Look in*** space select the Development directory in `E:\EUROPLEXUS`; in the ***File name*** field type `main.f` (or select ***Fortran files*** from the ***Files of type*** drop-down menu, then click on `main.f`) and click on ***OK***.

The file `main.f` is added to the `eplex` project Source Files entry.

9. Next, we set the project settings for the `eplex` project. Click on ***Project → Set Active Project*** and choose `eplex`. Click on the eplex files entry to highlight it. Click on ***Build → Set Active Configuration*** and choose ***eplex - Win32 Debug***.

   Then click on ***Project → Settings*** and choose the ***Fortran*** tab. In the ***Category*** field choose ***Runtime***, and then activate the ***Generate Traceback Information*** box. Click on ***OK***.

   In the same dialog, make sure to disable the box ***Array & String Bounds*** under ***Runtime error checking*** (this is necessary due to the variable passing strategy of EUROPLEXUS coming from its ancestors). Click on ***OK***.

   In the ***Fortran*** tab under ***Project → Settings*** choose ***Category → Fortran Data*** and in the ***Data Options*** menu, activate the box ***Variables Default to AUTOMATIC***. Click on ***OK***.

   In the ***Fortran*** tab under ***Project → Settings*** choose ***Category → Preprocessor*** and in the ***Include and USE Paths*** field, type in the following:

   ```
   \\SM48\E\EUROPLEXUS\Include;\\SM48\E\EUROPLEXUS\Module
   ```

   This has as an effect that include files (`*.inc`) and object module files (`*.mod`) which are not found on the current directory are searched in the two listed directories before the default directories.

   Note that the default directories may be inspected/set under ***Tools → Options → Directories → Show Directories for***. However, directories set in that way are valid for Developer Studio as a whole (i.e., for all projects) and not only for the current project. Therefore, it is advised to leave them at the default values.

   In the ***Link*** tab under ***Project → Settings*** choose ***Category → General*** and in the ***Object/Library modules*** field, insert, after the entry `kernel32.lib` and before `user32.lib`, the two EUROPLEXUS libraries `Libplex.lib Libplex_c.lib` (note that library names must be separated by a space) and the OpenGl related libraries `f90gl.lib f90glu.lib f90glut.lib bmplib.lib`. Make sure also that the system library `gdi32.lib` is included in the list (if not, add it as well).

   This sets the library search path for the current project only. Similarly to includes, library paths may be set at the general Developer Studio level (i.e., affecting all projects) in the ***Tools → Options → Directories → Show Directories for*** dialog, but this is impractical and strongly discouraged.

   In the ***Link*** tab under ***Project → Settings*** choose ***Category → Input*** and in the ***Additional Library Path*** field, type `\\SM48\E\EUROPLEXUS\Library`. This path is added to the default paths for library search and is used to locate the two EUROPLEXUS libraries mentioned in the previous step.

10. In the ***Link*** tab under ***Project → Settings*** choose ***Category → Input*** and in the ***Ignore Libraries*** field, type `libc.lib` after `dfconsol.lib`, separating it by a comma and no spaces (i.e., the entry should read: `dfconsol.lib,libc.lib`). This is needed because otherwise at link errors related to duplicated symbols appear (some symbols are defined both in `libcmt.lib`, which is used by default in QuickWin applications, and also in `libc.lib`, which is also used by default) due to a bug in the Developer Studio. This step is not needed in the QuickWin case.

11. The setting of the Workspace is now complete. To test it, first remove the filtered file `main.f` from the `\\SMNT48\E\EUROPLEXUS\Development` directory (to force its re-generation), then click on "Workspace plex: 2 projects" to highlight the whole workspace, and finally choose ***Build → Rebuild All***. This should rebuild the entire workspace, i.e. the two projects `devel` (filtering of `main.ff`) and `eplex` (compilation of `main.f` and link). The messages should be something like:

```
Deleting intermediate files and output files for project 'devel - Win32 Debug'.
Deleting intermediate files and output files for project 'eplex - Win32 Debug'.
--------------------Configuration: devel - Win32 Debug--------------------
Performing Custom Build Step on .\main.ff
Filtering .\main.ff
1
Filtering done
--------------------Configuration: eplex - Win32 Debug--------------------
Compiling Fortran...
E:\EUROPLEXUS\Development\main.f
Linking...

eplex.exe - 0 error(s), 0 warning(s)
```

A new executable `eplex.exe` should be produced in directory `debug`. This directory will also contain the object file `main.obj`, and a few auxiliary files (`Df60.pdb`, `eplex.pdb`). Two log files `devel.plg` and `eplex.plg`, one for each re-built project, appear in directory `E:\EUROPLEXUS\Development`.

8.2 Using the workspace

Once prepared an EUROPLEXUS development workspace as detailed in the previous Section (or, more efficiently, by using the command presented in Section 8.3), its use generally consists in adding new or modified source files to the `devel` and `eplex` projects, in re-generating the workspace by performing the necessary filterings, compilations and link, and in running the test code through the debugger.

It is essential to note that any modifications to source code should be exclusively performed in the unfiltered sources, i.e. to files `.ff`. The filtered versions of these files (`.f`) are to be considered as intermediate files that should never be edited by hand.

Adding a source file to the workspace consists in the following steps:

- Edit the file from scratch if it is new, else retrieve the current version of the file from the EUROPLEXUS source library (by the `epx_get` command) and edit it to perform the necessary modifications. In any case, a new file, say `celem.ff`, appears in the development directory.

- Add the `.ff` file to the devel project: click on the Source Files entry under devel files, press the right mouse button and click on ***Add Files to Folder***. Under ***File name*** choose the new file (`celem.ff`) and the click on ***OK***. The new file name appears under the Source Files entry.

- Set the Custom Build action for the `celem.ff` file. Click the `celem.ff` icon under Source Files in the devel files, click on ***Settings*** and choose the ***Custom Build*** tag. The ***Commands*** and ***Output*** fields are empty. The values from any previously set file must be copied: in the present example, the only available file is `main.ff`.

  Click on the `main.ff` icon in the ***Project Settings*** dialog, highlight and copy (`CTRL+C`) the contents of the ***Outputs*** field first. Then click on the `celem.ff` icon, paste the contents into the ***Outputs*** field (`CTRL+V`) and click on ***OK***. Repeat the same sequence for the ***Commands*** field. Note that it is essential to copy the ***Outputs*** field first, because otherwise an error message "Output field required" appears when clicking ***OK*** after the first paste.

  When performing the above settings, make sure that in the ***Project Settings*** dialog, the ***Settings For*** field contains either All Configurations or Win32 Debug (if this is the only configuration). In this way, the (same) filtering command will be active for all configurations.

- Filter the `celem.ff` file to produce `celem.f`. This may be accomplished e.g. by right-clicking on the `celem.ff` icon under the Source Files entry of devel files, and then by clicking on ***Compile celem.ff***.

- Add the filtered file `celem.f` to the eplex project. Click on the Source Files entry of eplex files, press the right mouse button and click on ***Add Files to Folder***. Select the `celem.f` file and click ***OK***.

- Re-build the `eplex` project. Click on the eplex files icon (make sure it is the active project), then click on ***Build → Build eplex.exe*** (or press `F7`). The compilation of `celem.ff` followed by a link should occur (no filtering is done because the filtered version of `celem` should be up to date).

8.3 Generating the workspace automatically

A perl script `epx_init` has been prepared, that automatically creates a new EUROPLEXUS development workspace in the current directory. This command is obviously available to all users and developers of EUROPLEXUS. The syntax is:

```
epx_init [-w] [-e] [-d] [-?]
```

where:

`-w`	Build a QuickWin application workspace instead of a Console application workspace.
`-e`	Produce an empty workspace. By default, the script inserts automatically into the workspace all `*.ff` files which are found on the current directory. However, the procedure to do so depends on the version of the Fortran compiler/Visual Studio installed. Therefore, if the installed software is updated, using the –e switch may be useful until the whole procedure has been updated.
`-d`	Delete any previous version of the workspace present on the current directory and build a fresh workspace. By default, if a previous workspace is already found on the current directory, an error message is issued.
`-?`	Prints short help with purpose and syntax of this command.

The command copies from directory `%EUROPLEXUS%\init` (or from `%EUROPLEXUS%\init_quickwin` , in case the `-w` switch is used) to the current directory a Developer Studio workspace named plex (consisting of files `plex.dsw`, `plex.ncb` and `plex.opt`), and containing the two projects devel (file `devel.dsp`) and eplex (file `eplex.dsp`). If any of these files exists already in the current directory, an error message is issued and the copy actions are not performed.

The projects are pre-set in order to contain the single source file `main.ff` (`main.f` in eplex). This source file is copied to the current directory from the EUROPLEXUS library, via the `epx_get` command, so as to be sure to get the latest version. In case the `-w` switch is used, the additional files `m_qwin.ff` and `ifwin.ff` are also contained in the project, since these are the only files of EUROPLEXUS that contain QuickWin-related code.

The `debug` subdirectory is <u>not</u> copied, since it is automatically generated when the workspace is re-built. The `main.f` file is also generated automatically from `main.ff` upon re-building.

It is advised to re-build the workspace immediately after its generation by the `epx_init` command, i.e. before adding any files into it. To rebuild the workspace, open file `plex.dsw` by Developer Studio (right click and then choose ***Open With MSDev***), click on ***Build → Rebuild All***, then click on ***File → Save All***. The following messages typically appear:

```
Deleting intermediate files and output files for project 'devel - Win32 Debug'.
Deleting intermediate files and output files for project 'eplex - Win32 Debug'.
Error scanning file E:\EUROPLEXUS\Essai\main.f for dependencies.
Build : warning : failed to (or don't know how to) build 'E:\EUROPLEXUS\Essai\main.f'
--------------------Configuration: devel - Win32 Debug--------------------
```

```
Performing Custom Build Step on .\main.ff
Filtering .\main.ff
1
Filtering done
--------------------Configuration: eplex - Win32 Debug--------------------
Compiling Fortran...
E:\EUROPLEXUS\Essai\main.f
Linking...

eplex.exe - 0 error(s), 0 warning(s)
```

The error message in scanning `main.f` for dependencies and the following warning message may be safely ignored.

The workspace is now ready to include new or modified files (as explained in Section 8.2) for the development and debugging of EUROPLEXUS.

All script files used in the interactive development and debugging of EUROPLEXUS are listed in the Appendix at the end of the present document. They are also summarized in Table 8.

Table 8 - Scripts for interactive code development

Scope	Syntax
Perform filtering as Custom Build action (used in Developer Studio projects only)	`[epx_filbat.bat]`
Perform filtering as Custom Build action (QuickWin version)	`[epx_filbat_quickwin.bat]`
Generate workspace	`epx_init [-w]`

8.4 Code profiling

It may be useful at times to profile code execution in order to obtain an estimate of the CPU time spent in the different subroutines of the program, in view of optimising calculation performance. This can be done, under MS Windows, in one of two ways: (i) from the command line, or (ii) under the Developer Studio graphical environment. We shortly describe hereafter the first method.

Profiling requires using three additional commands that are made available when installing the compiling environment: `prep`, `profile` and `list`. These admit several switches (for a complete list type for example `prep /?`) and hereafter only a minimal set, the one used for probably the simplest operations, will be discussed.

Assuming that one wants to check the CPU time spent in the various subroutines of the program, here is how to proceed:

1. Create a new, empty work directory and make a local copy of the routines under investigation, e.g. by the `epx_get` command.

2. Compile the routines by `epx_cmp`. Use optimized compilation (`epx_cmp -o`) preferably:, because else the ratios between CPU times spent in the various routines could deviate much from the behaviour of the standard (optimised) load module. Note, however, that debug-type compilation (`-w` switch in place of `-o`) is mandatory if line-by-line profiling is desired, see below. This will produce an auxiliary file `df60.pdb`.

3. Link the obtained object files with the standard EUROPLEXUS library by the command `epx_lk -w -o -p`. Note use of the `-p` switch: this prepares the load module `epx.exe` for profiling and produces an additional auxiliary file `epx.map`.

4. Call the `prep` command for the first time to prepare for profiling operations. For example: `prep /om /ft /sf _EUROPLEUXUS epx`. In this example the `/om` switch forces

prep to store the modified executable in a separate file, that will be named epx._xe. In addition, auxiliary files epx.pbi and epx.pbt are always produced. The /ft switch causes profiling by function with times and counts. The /sf switch defines the starting function for the profiler. The switch must be followed by the name of a function as listed in the "map file" (epx.map in our case). Spelling and letter case is significant. If this switch is not specified, the profiler starts with the highest-level routines: unfortunately, in the case of a QuickWin application like EUROPLEXUS, this has as an effect (profiler bug ?) that the real EUROPLEXUS routines (e.g. PRINC, CALCUL, CELEM etc.) are *not* listed in the results. Also, note that not all names listed in the .map file appear to work as starting points. For example, one finds three distinct names that coincide with the same object file (main.obj): EUROPLEXUS@0, _MAIN__ and _EUROPLEXUS. Of these, only the third one has the desired effect! Alternatively, one could start profiling at PRINC (/sf _PRINC@4) or even at CALCUL (/sf _CALCUL@84). The numbers at the end of the file names (4, 84) are the lengths of the variables in exchange list, and are therefore subjected to change: always take these names from the .map file of the version you are trying to profile. Another example could be the case that one wants to verify the CPU time spent in just one or a few routines: prep /om /ft /excall /inc calcul.obj /inc celem.obj epx. The /excall switch excludes every function/source/obj/lib from profile. The /inc switch followed by a name (e.g. calcul.obj) includes the calcul subroutine in the profile. Thus, in the above example two (local) subroutines would be profiled: calcul and celem. Note, however, that in this case all CPU time figures reported in the profile results will refer to the highest-level routine as 100% of the time spent, but this may of course be much less than the total time spent in the calculation.

5. Run the code under the profiler by the profile command. For example: profile epx bm_str_eled01, assuming that the test case bm_str_eled01 is the one of your choice. Note that the file bm_str_eled01.epx (and if necessary also the corresponding mesh file bm_str_eled01.msh) must be present in the local directory, because the call to epx is done directly, and not via the epx_bench command for example. Expect much slower execution of the test run as compared to normal execution, because profiling consumes quite a lot of CPU time in itself. This generates an extra auxiliary file, epx.pbo. Unfortunately, running times under the profiler may be 10 to 50 *times* longer than without profiler (depending upon the starting point of profiling and the number of routines profiled).

6. Call the prep command for the second time to conclude profiling operations. A typical command would be prep /m epx. The /m switch tells prep to merge the pbt and pbo files. This modifies the auxiliary file epx.pbt.

7. Finally, produce a listing of the profile results by calling plist. For example: plist epx >bm_str_eled01.prof. This command would produce a listing of profiler results on file bm_str_eled01.prof. By default (or by using the /st switch) the list is sorted by function time. Alternatively, the /sc switch sorts results by hit/sample count.

Note that in step 4 above, failing to specify both the /sf and the /excall switches has the following, probably undesired, effect: all routines are considered for profiling, included system and Fortran library routines. As a result, the profiler shows the times related to some "exotic" routines, but *not* the one spent in the routines of interest here (calcul and celem). These are probably cumulated and hidden in the system routine _WinMain (apparently a QuickWin routine) that is usually reported as the one that uses most of the CPU time.

By using /excall followed by one or more /inc switches like in the above example, computes and shows only the time spent in the specified routines. Note that this time is therefore usually lower than the total CPU time spent in the process.

An important observation is that repeated use of the profiler in the same directory must be used with care: in fact, the results may be *cumulated* over different runs rather than updated each time a new run is performed. This may be verified in the listing produced by `plist`: there must be only one line reading "Command line …". To avoid cumulation of results, always remove the `.pbi`, `.pbo`, `.pbt` files (and also `._xe`, for security) before repeating a profile operation:

```
rm epx.pb* epx._xe
prep /om /ft …
profile epx …
prep /m epx
plist epx >…
```

8.5 Using the LIB command

The `LIB` command may be used to deal with object libraries. For a list of all available switches, type `LIB /?` at the command prompt. Here is a list of commonly used ones:

- To create a new library `name.lib` containing all object files present in the current directory, type: `LIB /OUT:name.lib *.obj`.
- To list the contents of a library `name.lib`, type: `LIB /LIST name.lib`.
- To extract a member, say `main.obj`, from a library `name.lib` (note that the member is not deleted from the library), type: `LIB /EXTRACT:main.obj name.lib`.
- To delete a member, say `main.obj`, from a library `name.lib`, type: `LIB /REMOVE:main.obj name.lib`.
- To replace a member, say `main.obj`, in a library `name.lib`, type: `LIB name.lib main.obj`.
- To add a new member, say `main.obj`, to a library `name.lib`, type: `LIB name.lib main.obj`.

Note that the command syntax to replace a member or to add a new member is the same. In other words, if the member already exists, it is replaced (without prompting), else it is added.

8.6 Modifying the code stack size

In Fortran, variables that are not statically or dynamically allocated are placed on the stack. This allocation is efficient, but the amount of stack space is limited. Under Windows, the defailt stack size is 1 Mbyte, which may be too small for some large custom applications.

While the EUROPLEXUS program is being restructured to gradually replace stack variables with dynamically allocated ones (this method is safer and does not seem to be much less efficient) it may occasionally happen that a user receives a runtime error message related to insufficient stack space.

In this case, there are several possibilities to increment the stack size, as described hereafter:

- To set the stack size in Microsoft Developer Studio (i.e. in the interactive development environment decribed in Sections 8.1 and 8.2), select the eplex project, then click on ***Project*** → ***Settings***, choose the ***Link*** tab and under Category ***Output*** in the Stack allocations, Reserve field enter the size of the stack expressed in hexadecimal digits. For example, to set a 100 Mbyte stack type `/stack:0x6400000`.
- To set the stack size when performing the link from the command line, add the following switch to the link command (usually executed via the compilation command `df`): `df …`

`/LINK … /STACK:reserve`, where `reserve` has the same meaning as above. This is now done via the `-s` switch of the `epx_lk` command.

- Note that it is possible to modify the stack size of an executable (e.g. `europlexus.exe`). This is accomplished by the `EDITBIN` command. The syntax is: `EDITBIN /STACK:reserve name.exe`. Here `reserve` has the same meaning as above and `name.exe` is the name of the executable to be modified.

9 Summary of used tools

For ease of reference, the following Table summarizes all commercial and free tools that are used in the management of the EUROPLEXUS mirror site at JRC.

Table 9 – Overview of tools used at the JRC mirror site

Purpose	Name	Type/Location
Compilation	F90 Compiler: COMPAQ Visual Fortran Professional Edition 6.6	Commercial
Compilation	C Compiler: MicroSoft Visual C++ 6.0	Commercial
Scripting	Perl	http://www.activestate.com
Mailing	Sendmail perl package, to send e-mail from a perl script	http://www.perl.com/CPAN-local/modules/index.html
Scripting	NT native port of Unix commands (GNU)	http://www.weihwnstephan.de/~syring/win32/UnxUtils.html
Editing	NT port of the `vi` editor	http://www.vim.org
LaTeX compiler	MiKTeX	http://www.miktex.org
PostScript visualizer	Ghostscript/GSview	http://www.cs.wisc.edu/~ghost/
PostScript to vector format filter	Pstoedit[2]	http://www.pstoedit.net/pstoedit
LaTeX to HTML translator	Tth	http://hutchinson.belmont.ma.us/tth/
LaTeX to HTML translator	Hevea/Hacha (Home page, documentation)	http://para.inria.fr/~maranget/hevea/
LaTeX to HTML translator	Hevea/Hacha (NT port)	http://www.arch.ohio-state.edu/crp/faculty/pviton/support/
Custom headers/footers	`fancyhdr.sty`[3]	http://tug2.cs.umb.edu/ctan/
Current time	`time.sty`	http://tug2.cs.umb.edu/ctan/
Comments	`comment.sty`	http://tug2.cs.umb.edu/ctan/

10 Implementation notes

This Section collects various implementation notes that help tracking the evolution of the mirror site and of the used tools.

[2] This package is now included in the distribution of GSview.

[3] Note that this package was non-standard only in previous versions of MiKTeX. It is contained by default in the "large" installation of version 2.1.

10.1 Version March 2001

In February-March 2001 the mirror site at JRC was completely updated. Most of the tools used were updated to the current version:

- The C compiler is still Version C++ 6.0, and the Fortran compiler is still Version 6.1.
- The `perl` language is now Version 5.6.0.618. This required a small modification in an old script (`searchfc.pl`, used only in the PLEXIS-3C project) that used the `glob` command. The file names starting with `\\` must be protected by doubling the backslashes (`\\\\`) before doing the globbing.
- The `Sendmail` perl package was not updated.
- The NT native port of Unix commands (GNU) was updated. This apparently corrected some misfunctioning that had been occasionally observed with the `diff` command.
- The `vim` editor was updated to Version 5.7, and syntax highlighting was enabled at the installation. A small customization was needed so that the Fortran syntax highlighting is applied also to files with the suffix `.ff`, which is not recognized by default as a valid Fortran file extension. This involves files `D:\Appl\Vim\Vim57\filetype.vim` (add `*.ff` to the Fortran extensions) and `D:\Appl\Vim\Vim57\syntax\Fortran.vim`. Similarly, to make `.ttx` files treated like TeX (`.tex`) files, add `*.ttx` to the TeX extensions in file `D:\Appl\Vim\Vim57\filetype.vim` (apparently, the file `D:\Appl\Vim\Vim57\syntax\tex.vim` needs no modification in this respect).
- The MiKTeX package was updated to Version 2.0. Since the new package no longer contains the `tth` utility, the old version of `tth` was kept (it still works). The additional style packages were not updated.
- The Hevea package was updated to Version 1.05. Under some circumstances, a new style package `comment.sty` is now necessary, see Section 7.5.9. This file was apparently hardcoded within `hevea.sty` in previous versions of Hevea, but now it should be present as a separate file, that is called from `hevea.sty`. Apparently, this file is not used when processing a file that includes `hevea.sty` by `hevea` itself, but it is necessary when processing that file with LaTeX or PDFLaTeX.
- The GhostScript/GSview packages were updated to Versions 6.50 and 3.6, respectively.
- The Pstoedit package was updated to Version 3.21.

10.1.1 Slow QuickWin text display

It was noted that the display of text lines in the text window of EUROPLEXUS under QuickWin became extremely slow. The machine seemed to access the disk at every displayed line, and this rendered program execution even slower than on the previous machine, despite the higher CPU clock and better overall performance of the new machine.

It was discovered that the problem is due to the setting of the graphics card (Matrox). The problem arises when “True Color at 32 bit” is chosen as display setting, and with a resolution of 1600 x 1200.

If the setting “True Color 24 bit” (16 Million colors) is used at 1600 x 1200, the problem almost disappears. Using less colors and lower resolutions speeds up even more the QuickWin text display, but the more drastical difference is between 24-bit color (slow but acceptable) and 32-bit color (totally unacceptable).

10.2 Compilation problems under version 6.5A

The Fortran compiler (from COMPAQ) was upgraded on 25 June 2001 to Version 6.5A. This corresponds to Version 6.5 (an upgrade to 6.1) to which the upgrade to 6.5A, that can be freely downloaded from Compaq's site, has been applied.

Two problems were discovered, which are briefly decribed hereafter.

10.2.1 Performance degradation in some domain decomposition test cases

This problem is not peculiar to Version 6.5A but it existed also with previous versions. When running some mid-size simulations involving domain decomposition (a model recently implemented in EUROPLEXUS) some severe performance degradation is noticed with respect to small-size tests that use the same model (e.g., the standard benchmark tests).

After investigation, it was found that this is due to hybrid calls between the new F90 data structure used to implement the domain decomposition data structures and old F77 routines. Most of the slow-down occurs in routine D_LOOPELM, in the calls to D_ELCARA, WPGEN, WPCOP and to the element routines (e.g. CUBE).

The performance degradation may be impressive: for a test case with 2000 elements the code is 100 times slower than expected. This effect is not observed on Unix machines, where the CPU time is as expected roughly proportional to the number of elements.

The problem is due to calls that pass F90 `POINTER` arrays to a F77 routine, where these arrays are declared as assumed-size ones, i.e. with the obsolescent "`(*)`" dimensioning declaration. In the calling routines, the POINTER arrays are usually components of F90 derived types.

Consider the following example:

```
TYPE nodvar
  REAL(8), POINTER :: var
END TYPE nodvar
. . .
TYPE(nodvar) :: v
. . .
CALL sub1 (v%var)
. . .
SUBROUTINE SUB1 (vel)
REAL(8) vel(*)
. . .
```

A derived type `nodvar` contains a deferred-shape (`POINTER`) array `var`. Such array may be *non-contiguous*, because for example one may declare a stride, like in `var(1:10:2)`. When passing such an array to subroutine `SUB1`, which is an old F77 subroutine whose interface is *not* known to the calling program, and where `v` is declared as an assumed-size array, the compiler must provide a contiguous array since the `SUB1` routine expects contiguous data. Therefore, it copies `v%var` on a temporary, contiguous array, passes the address of this array to `SUB1`, and on return it copies back the data from the temporary array (that may have been modified in `SUB1`) to `v%var`. This of course is very expensive, especially when the size of the array is large.

Some compilers avoid the double copy by checking at run time whether the copy is actually necessary or not. Compaq intends to implement such a strategy in the next publicly available upgrade, due in August 2001. For this reason, no intervention in the code is planned since the next compiler version should fix the problem.

Alternatively, one could of course fix the problem by passing `v` instead of `v%var`, and by doing the pointing in `SUB1`:

```
TYPE nodvar
  REAL(8), POINTER :: var
END TYPE nodvar
. . .
TYPE(nodvar) :: v
. . .
CALL sub1 (v)
. . .
SUBROUTINE SUB1 (v)
TYPE(nodvar) :: v
REAL(8), POINTER :: vel
vel => v%var
. . .
```

However, this involves modifying all called routines, in particular all element routines, which is undesirable at this stage of the EUROPLEXUS project.

In the long term, when all old data structures and subroutines will be replaced by more modern F90 code, such potential problems with hybrid code will gradually disappear.

10.2.2 BACKSPACE failure

A problem appeared in benchmark tests that produce unformatted K2000 output files to be post-processed by CASTEM 2000. In such files, at each requested time station the output data are stored, and then an "`ENREGISTREMENT DE TYPE 5`" is written, that flags the end of data for CASTEM 2000. At the next time station, a `BACKSPACE` is performed to remove this record from the file before writing the next set of data.

When the data is written as formatted, everything is correct, but when an unformatted write is performed the subsequent `BACKSPACE` fails. The code asks for input data from the keyboard, then it fails.

Finally, this problem manifests itself only in the QuickWin version of the executable, not in the Console version.

The problem was found to be due to a wrong component in the runtime system library `dformt.lib`. A patch file `for_backspace.obj` was received from Compaq, which fixes the problem. This file may either be explicitly linked in when producing an executable from the command line, or inserted in a Developer Studio project together with the other "`.f`" files. Note that this file is only used for QuickWin applications, and not for Console applications (that use `dfor.lib` instead, containing a correct version of the `BACKSPACE` command).

A better solution is perhaps to modify the system library `dformt.lib`, by replacing the wrong object file. The commands to do this are:

```
COPY dformt.lib dformt_ori.lib
LIB /REMOVE:for_backspace.obj dformt.lib
LIB dformt.lib for_backspace.obj
```

This commands patches the library by replacing the faulty component object file.

After applying this patch it was discovered that under some circumstances (e.g. during the normal evolution process spawned by `epx_evol_start`) the problem reappeared. The problem could be avoided by patching also the other library (`dfor.lib`). The reason why this works is unknown, because the QuickWin version used for the normal evolution process should use only `dformt.lib`. Anyway, the second library was patched by the commands:

```
COPY dfor.lib dfor_ori.lib
```

```
LIB /REMOVE:for_backspace.obj dfor.lib
LIB dfor.lib for_backspace.obj
```

With these corrections all code versions and evolution procedures seem to work correctly.

10.3 Version September 2001

In September 2001 a new version of the Fortran compiler was installed: Compaq Visual Fortran 6.6. It is a free upgrade of Version 6.5A, itself a free upgrade to major Version 6.5. This version fixes a serious problem that had been encountered in some tests that use the multidomain model (see 10.2.1).

In this model, new complex data types are defined using the Fortran 90 `TYPE` definition statement. Some of these types include large `POINTER` arrays. Then, old Fortran 77 routines are called by passing these arrays in exchange list. The old version of the compiler performed a double copy of the arrays (on routine call and on return) to cope with the possibility (in Fortran 90) to pass a non-contiguous array. This severely degraded the performance: the conde under Windows NT was up to 100 times slower than on a comparable Unix machine. The same test, passed without sub-domains, ran with the expected efficiency.

The problem could be avoided by either modifying the called routines, or by using explicit interfaces, but this was felt to be too laborious in the present intermediate stage of Fortran 77 to Fortran 90 conversion of the code. The new version of the compiler fixes the problem because it is able to recognize, at run time, that the passed array is indeed contiguous so that the double copy is unnecessary.

Another improvement introduced in this period was the filtering of the User's manual. Now the manual source files have the extension `.ttx`, and the LaTeX input files `.tex` are obtained by filtering via the standard `epx_filter` program, in analogy with what is done with the Fortran source files. This mechanism allows to cope with platform dependencies (the LaTeX version in use at the various sites are somewhat different), and also to deal with possible reserved parts of the manual (connaissances métiers).

10.4 Compilation problems under version 6.6

The Fortran compiler (from COMPAQ) was upgraded on 30 August 2001 to Version 6.6, a free upgrade for 6.5 users that may be obtained from the Compaq site.

The problem with `BACKSPACE` affecting the previous version (see Section 10.2.2) persists but Compaq will correct it in the next version. Therefore, the same patches described above for the previous version had to be applied. The same object file `for_backspace.obj` as before was used, since Compaq informed us that it is compatible with both versions of the compiler.

The good news is that, as had been promised by Compaq, the severe performance degradation problem mentioned in Section 10.2.1 completely disappeared.

No additional compiler problems have emerged to date on this version.

10.5 Version December 2001

On December 2001 the operating system was changed from Windows NT to Windows 2000 Professional. All the public-domain tools were updated to the most recent available versions. Here is a short list of the settings that have been necessary.

10.5.1 Perl

The new version 5.6.1 automatically performs file association when installed, thus the manual setting described in Section 3.1.1 is no longer necessary. However, the PATHEXT variable setting (Section 3.1.2) is still required.

A small incompatibility was found concerning the globbing (expansion of file names). A perl statement such as:

```
@FileNames = glob ("$epxbfiles\*.epx");
```

no longer works. This is thought to be due to the simultaneous presence of backslashes (\) and wildcards (*). To circumvent the problem, all backslashes in the directory name ($epxbfiles) are changed to forward slashes:

```
$epxbfiles =~ s/\\/\//g;
$epxbfiles = "$epxbfiles/*.epx";
@FileNames = glob ("$epxbfiles");
```

The Date package (file `C:\perl\site\lib\HTTP\Date.pm`) has been customized according to the instructions in Section 3.6.

10.5.2 Vim

The new version 6.0 of the Vim editor introduced some configuration problems. When installed with all default settings, the behaviour of the editor was not the expected one especially as far as mouse support in copy and paste operations was concerned.

In the previous version, selecting text (by simply highlighting it) with the left mouse button copied the selected text to the clipboard. The text could then be pasted to a different place (in the same or even in another Vim window) by simply pressing the central mouse button.

In the new version, the same operation could be performed only by a `CTRL-C` followed by `CTRL-V` , which is slightly less comfortable.

To restore the old behaviour, and to adjust other less important settings, a configuration file `_vimrc` has been prepared. This file has to be placed in the Vim directory (`C:\Vim` in the current implementation). The file has been obtained by editing and customiziong the file:

```
C:\Vim\Vim60\vimrc_example.vim
```

Here is the contents of the file:

```
" An example for a vimrc file.
"
" Maintainer:   Bram Moolenaar <Bram@vim.org>
" Last change:  2001 Jul 18
"
" To use it, copy it to
"     for Unix and OS/2:  ~/.vimrc
"       for Amiga:  s:.vimrc
"  for MS-DOS and Win32:  $VIM\_vimrc
"    for OpenVMS:  sys$login:.vimrc

" When started as "evim", evim.vim will already have done these settings.
If v:progname =~? "evim"
  finish
endif

" Use Vim settings, rather then Vi settings (much better!).
" This must be first, because it changes other options as a side effect.
Set nocompatible

" allow backspacing over everything in insert mode
set backspace=indent,eol,start

"FC set autoindent              " always set autoindenting on
"FC if has("vms")
"FC   set nobackup      " do not keep a backup file, use versions instead
"FC else
"FC   set backup        " keep a backup file
```

```
"FC endif
set noautoindent
set nobackup
"FC
set history=50          " keep 50 lines of command line history
set ruler               " show the cursor position all the time
set showcmd             " display incomplete commands
set incsearch           " do incremental searching

" For Win32 GUI: remove 't' flag from 'guioptions': no tearoff menu entries
" let &guioptions = substitute(&guioptions, "t", "", "g")

"FC
set clipboard^=autoselect
set guioptions+=a
"FC

" Don't use Ex mode, use Q for formatting
map Q gq

" Make p in Visual mode replace the selected text with the "" register.
Vnoremap p <Esc>:let current_reg = @"<CR>gvs<C-R>=current_reg<CR><Esc>

" This is an alternative that also works in block mode, but the deleted
" text is lost and it only works for putting the current register.
"vnoremap p "_dp

" Switch syntax highlighting on, when the terminal has colors
" Also switch on highlighting the last used search pattern.
If &t_Co > 2 || has("gui_running")
  syntax on
  set hlsearch
endif

" Only do this part when compiled with support for autocommands.
If has("autocmd")

  " Enable file type detection.
  " Use the default filetype settings, so that mail gets 'tw' set to 72,
  " 'cindent' is on in C files, etc.
  " Also load indent files, to automatically do language-dependent indenting.
"FC  filetype plugin indent on

  " For all text files set 'textwidth' to 78 characters.
"FC  autocmd FileType text setlocal textwidth=78

  " When editing a file, always jump to the last known cursor position.
  " Don't do it when the position is invalid or when inside an event handler
  " (happens when dropping a file on gvim).
"FC  autocmd BufReadPost *
"FC    \ if line("'\"") > 0 && line("'\"") <= line("$") |
"FC    \   exe "normal g`\"" |
"FC    \ endif

endif " has("autocmd")
```

Lines highlighted with "FC (or contained within a couple of such commented lines) have been customized. Autoindenting is disabled, and no automatic file backup is set. The lines starting with `set clipboard` and `set guioption` are those which have to do with mouse copy and paste functions. Automatic filetype recognition has been disabled (however, syntax highlighting works nevertheless) and automatic placing of the cursor at the last previous position when re-opening a file has been also avoided (this may be confusing when editing large files).

Another useful configuration concerns the use of the Vim editor from the command line. To start the program from any console window, a simple script `vi.bat` has been prepared and placed on the `PATH`:

```
@echo off
start gvim.exe %*
```

The `start` command executes Vim in a window by returning the prompt to the console from which the command has been issued. In this way the user may continue to type other commands in the same console while at the seme time editing in the Vim window. For example, multiple Vim windows may be generated from the same console window, by typing: `vi coco.txt`, then `vi celem.ff` etc.

10.5.3 MiKTeX

The current MiKTeX package (Version 2.1) uses an installer `setup.exe` that is used to download, install and continuously update the various packages that compose the installation. A so-called "large" system installation was chosen (about 100 MB on disk).

After installation, it is necessary to perform the file association of `.dvi` files with the `yap` previewer (`C:\texmf\miktex\bin\yap.exe`) by hand (***Tools → Folder options***). Do it as administrator to have it automatically also for the other users.

10.5.4 Hevea

The hevea package is still at version 1.05, at least in the Windows port, which is the one needed here. To install the package:

- Create a directory `C:\Hevea`.
- Unpack the distribution (file `winport.zip`) to the above mentioned directory.
- Move or copy the `hevea.sty` file from `C:\Hevea` to the customized MiKTeX folder `C:\texmf\tex\latex\folco`.
- Copy the four batch files `hevea.bat`, `hacha.bat`, `hinfo.bat`, `htext.bat` to a place on your path (e.g. to `D:\Users\Folco\Run`). If the hevea package has been installed in `C:\Hevea` these four scripts need not be modified, else edit the `set heveadir` line in each of these scripts.
- Copy the hevea manual file `hevea-1.05-manual.pdf` to a newly created directory `C:\texmf\doc\hevea`.
- To test out the installation: go to directory `C:\Hevea\test` and try out: `hevea pavtest.tex` (compare the result `pavtest.html` with the file already present in that directory), and `htext pavtest.tex` (compare the result `pavtest.txt` with the file `pavtext.txs` already present in that directory). Small differences in the files are acceptable.

10.5.5 Tth

The tth package is no longer contained in the standard MiKTeX distribution. Use the old package. Installation consists in the following:

- Create a directory `C:\texmf\doc\tth` and copy to it the four files: `license.txt`, `tth.1.dvi`, `tth.gif` and `tth_manual.html`.
- Copy the executable `tth.exe` to `C:\texmf\miktex\bin`.

10.5.6 Ghostscript and GSview

The current versions are Ghostscript 7.03 (single self-extracting file `gs703w32.exe`) and GSView 4.1 (single self-extracting file `gsv41w32.exe`). Installation consists simply in double-clicking the self-extracting executable files and following the instructions.

By default, Ghostscript installs in `C:\Gs` while GSview installs in `C:\Ghostgum`.

Note that the distribution of GSview also includes PstoEdit, and this is automatically installed. To verify this, open a document with GSview, the click on ***Edit → Convert to vector format***. Many vector formats, including for example Windows Metafile (WMF) should be available. This allows to convert any PostScript image to Windows Metafile for inclusion, e.g., in a Word document.

A problem that has emerged and is still unresolved with GSview is that, when viewing on screen a file under the normal (folco) user, the quality of visualized fonts (e.g. for a standard FrameMaker-generated document) is far poorer than when the same document is viewed under the administrator account. This may perhaps depend on system (screen) settings, that vary from user to user, but the wrong setting (if any) is not known to date. Uninstalling the packages and re-installig them as folco rather than administrator did not solve the problem either.

10.5.7 PStoEdit

As mentioned in the previous Section, this is now included in GSview.

10.5.8 Installation of Adobe products

A series of (commercial) products from Adobe, whose installation and set-up is quite tricky, is heavily used at our site. Although these products are not part of the mirror site proper, some guidelines are nevertheless given for their correct installation and tuning.

Initialization

Before installing any Adobe products, it is best to uninstall any previously installed products of the same type, and closely follow the instructions below, in the given order. All installations should be performed under the administrator account.

Adobe Type Manager

First, install Adobe Type Manager 4.1. Insert the FrameMaker 6.0 distribution CD and navigate to `R:\English\ATM\Setup.exe`. Double-click on the executable and follow the instructions. This installation is usually smooth.

FrameMaker 6.0

Install a default PostScript printer on your system, if not already available. For example, the hp4000_ps printer. Choose it as your default printer.

Then, install FrameMaker 6.0. Insert the FrameMaker 6.0 distribution CD and navigate to `R:\FM\Setup.exe`. Double-click on the executable and follow the instructions. After installing FrameMaker proper, the installer proposes to install also Acrobat Distiller: click on `Cancel` (you will install it later separately from the Acrobat 4.0 CD). Next, it proposes to install the AdobePS driver: click on `Cancel`.

Next, update FrameMaker 6.0. The `FM60UPD.EXE` update available on Adobe's site should update version 6.0p357 to 6.0p405. Since the version you just installed is (check it!) already 6.0p405, there is no need to apply this upgrade.

Next, update the FrameMaker 6.0 on-line manuals by using `FMMANUAL.ZIP` update package available from Adobe. Unzip the three contained PDF files and replace them in the directory `C:\Program Files\Adobe\FrameMaker6.0\Onlinemanuals`. Restart your machine for safety.

A known bug of this installation is that a FrameMaker program group is created for the administator, but not for the other users. This may be easily fixed (better at the end of all installations), see below.

Acrobat 4.0

Next, install Acrobat 4.0 from the separate CD (not the version that comes on the FrameMaker 6.0 CD). Follow the instructions, the installation is smooth. At the end be sure to click on `Cancel` when the Adobe Registration panel appears. Restart the computer as indicated.

First, apply the update from 4.0 to 4.0.5, file `RS405ENG.EXE`. This apparently updates only Acrobat Reader (to Version 4.05c).

Finally, apply the update patch `AC405UP2.EXE`. At the end, the Acrobat version should (still) be Acrobat 4.0, the Acrobat Distiller version should be Distiller 4.0, and the reader version should be 4.05c.

It is important to note that this installation will produce two extra "printers" in your control panel settings, namely:

- Acrobat Distiller
- Acrobat PDFWriter

Neither of these will be actually used, but leave them for completeness (they are harmless).

After this installation you should have a programs group containing, among other things: Adobe Distiller 4.0, Adobe Acrobat 4.0 and Adobe Reader 4.0.

Adobe PostScript Driver

The installation of this component is extremely tricky. Unfortunately, it seeem absolutely necessary to install it, else the PostScript files produced by FrameMaker are often incompatible with both GSview and the distiller.

The installer, available from Adobe, is the executable file `WINSTENG.EXE`. It contains versions for Windows 98, Nt or 2000 and automatically recognizes the correct system upon installation. Under 2000, it should install a driver called Pscript5 5.2.

Before installing, it is necessary to get a PPD (PostScript Printer Description) file for the "printer" that will be used. In our case, we do not want a real printer, but a generic PostScript printer to print from FrameMaker to a file. This file will then be distilled by Acrobat to obtain the PDF version. The necessary PPD is therefore the one corresponding to the Acrobat Distiller printer.

Note that under NT it was not necessary to have this PPD file during the installation. In fact, one could install a so-called Generic PostScript printer, and then configure it (***Settings → Printers***, Right click on Generic PostScript Printer, choose ***Properties***: under ***Driver***, choose AdobePS Acrobat Distiller).

Under 2000 this *a posteriori* setting is no longer available, because the association with the Acrobat Distiller driver must be done during the installation of the Adobe PostScript Driver. The required PPD file was found (by chance) to be `ADIST4.PPD` (14565 bytes). It may be found under the installation directory of FrameMaker 6.0: `C:\Program Files\Adobe\FrameMaker6.0\ppds`.

Copy this file to a temporary place (for safety), then launch `WINSTENG.EXE`:

- Choose: Install a New PostScript printer;
- How is your Printer connected: choose Directly (Local printer);
- Local port selection: choose FILE;
- Select printer model: the only available one in the list is Generic PostScript Printer, that would install a file `DEFPRTR2. PPD`. This is wrong!. Click on Browse and locate the `ADIST4.PPD` (Acrobat Distiller) file instead;
- Choose Not Shared;
- Under Printer Name type: Generic PS Printer (distiller);

- Should this be your Default Printer: answer No;
- Should test page be printed: answer No;
- Configure: answer Yes, make sure the only available paper size is A4.

To test out your freshly installed driver, first make sure that the new "printer" named 'Generic PS Printer (distiller)' is available in your printers list. Then, open a FrameMaker document and try printing to this printer (actually, to a PostScript file through this driver). Visualize and check the PostScript file by GSview. Convert it to PDF by Acrobat Distiller.

The nasty points where all other drivers, more or less, seem to fail are the following:

- Mathematical formulas containing characters with a tilde under them (to represent a vector, matrix or tensor): the tildes tend to disappear!
- Some large symbols (summations for example) have a tendency to disappear.
- Verify the correct treatment of colors, they should be visible both in GSview and in the PDF file.

Note that when you choose to print to the new driver from FrameMaker, it may warn that some fonts may be substituted. This is usually not a major problem.

Illustrator

To install Adobe Illustrator 9.0, use the distribution CD. The installation runs smoothly. Then, there are two updates (available from Adobe) that can be applied (in this order).

The first update, `AI901ENG.EXE`, is from version 9.0 to 9.0.1.

The second update, `AI902ENG.EXE`, is from version 9.0.1 to 9.0.2.

Fixing FrameMaker program group

The FrameMaker program group is installed under the administrator, and not under all users. To fix this (as administrator), you must cut the `FrameMaker6.0` folder from `C:\Documents and Settings\administrator.SMCM\Start Menu\Programs` and paste it under `C:\Documents and Settings\All Users\Start Menu\Programs\Adobe`.

You may act similarly also to group all other Adobe applications (e.g., Illustrator) under the same program group.

Fixing file associations

After all these installations, file associations are probably not what one would like to have. This may be fixed (under administrator) and is automatically available to all users. The adjustments are done in Windows Explorer by clicking on ***Tools → Folder Options → File Types***, and by selecting the following file extensions: .PS (for PostScript files, note that this applies automatically also to .EPS, i.e. encapsulated PostScript files), and .PDF.

The suggested settings are:

- .PS file default action: `open`. Application: `"C:\Ghostgum\gsview\gsview32.exe" "%1"`. No DDE.
- .PS file action `print`. Application: `"C:\Ghostgum\gsview\gsview32.exe" /p "%1"`. No DDE.
- .PS file action `AI`. Application `"C:\Program Files\Adobe\Illustrator 9.0\Illustrator.exe" "%1"`. No DDE.

- .PS file action `PDF`. Application `"C:\Program Files\Adobe\Acrobat 4.0\Distillr\AcroDist.exe" %1`. No DDE.

- .PDF file default action: `open`. Application: `"C:\Program Files\Adobe\Acrobat 4.0\Acrobat\Acrobat.exe" %1`. No DDE.

- .PDF file action: `open with reader`. Application: `"C:\Program Files\Adobe\Acrobat 4.0\Reader\AcroRd32.exe" "%1"`. Use DDE. Dde Message: `[FileOpenEx("%1")]`. Application: `acroview`. DDE Application not running (leave empty). Topic: `control`.

- .PDF file action: `open with gsview`. Application: `"C:\Ghostgum\gsview\gsview32.exe" "%1"`. No DDE.

10.6 Using the standard executable

A problem may arise under Windows concerning the use of the standard executable file `europlexus.exe`. When a user launches a job that invokes this executable (which is the default behaviour of the `epx_bench` command, see Section 5.6), the system "locks" the file so that it may not be altered (removed, renamed or replaced) until the job is finished.

This causes a problem if the job is a long one, lasting perhaps several days or more, because meanwhile the automatic evolution procedure is unable to update the executable file.

To avoid this, the launching command (`epx_bench`) has been modified (January 2002) so that:

- If the user launches a local version of the executable (i.e. either uses the `epx_bench` command without the `-e` switch, or uses the `epx_test_benchmarks` command without the `-s` switch), then the previous behaviour is retained;

- If the user invokes the standard executable (via the `epx_bench -e` or the `epx_test_benchmarks -s` commands), then the current standard executable is first copied to a local directory (`%TEMP%`, which must therefore be defined in the user's environment) under a unique name, and then executed from that directory. When the run terminates, the local executable is removed, so that the next time the command is invoked a fresh (possibly more recent) version is taken.

The unique name strategy is adopted to avoid problems in the case that a user launches several partially overlapping runs at the same time. In this case, having a single local executable would be a problem because each job would try to replace it at the beginning of the run and or remove it at the end of the run. A more sophisticated strategy could be set up, but this would increase the possibility of failures.

The unique name solution allows to keep things simple. This name is automatically generated by appending to the executable base name the current date and time. A user launching several jobs at a time will have a corresponding number of (possibly identical) executables on his or her local directory.

In this way the standard executable remains always "unlocked" and the automatic evolution procedure may update it at any time.

The `epx_test_benchmarks` command has also been modified so that, if the user chooses the standard executable (`-s` switch), this executable is copied just once to the temporary directoy, then used to perform all tests, and finally automatically deleted by the procedure. This is to avoid that a separate executable copy is produced for each test, which would be a large waste of resources.

10.7 Version January 2003

On January 2003 the operating system was changed from Windows 2000 Professional SP3 to Windows XP Professional SP1. All the public-domain tools were updated to the most recent available versions. Here is a short list of the settings that have been necessary.

The name of the machine hosting the JRC mirror site has changed from `SM34` to `SM48`. All procedures containing this name hard-coded have been updated.

10.7.1 Operating system

The new version of the operating system is MS Windows XP Professional, Service Pack 1. The machine is a Pentium 4 at 2.0 GHz with 512 MB of RAM.

10.7.2 Developer Studio and Compilers

The installed compilers and programming environment are the same as under the previous installation:

- Microsoft Visual Studio 6.0 Professional Edition, with Service Pack 5.
- Visual C++ 6.0 Compiler
- Compaq (now Intel) Visual Fortran Professional Edition 6.6

As concerns the Fortran compiler, the installation proceeds in two steps: first install Version 6.5, then install the upgrade to 6.6.

The same patch as described in Section 10.2.2 to the Fortran compiler has been installed to avoid the problem with the `BACKSPACE` instruction.

All sources have been re-compiled and the modules and libraries re-built. The benchmark test suite (currently 245 tests) executes correctly with the following performance (see also Section 10.1.1):

- QuickWin version with screen setting at 1600x1200, 32-bit color: 20 minutes elapsed time
- Same but with 16-bit color: 15 minutes
- Batch version (no QuickWIn windows): 13 minutes

It is therefore chosen to set the screen at 16-bit color.

10.7.3 Animations quality problem

A problem concerning the quality of animations played on the new machine/video platform by the Windows Media Player has been detected. The quality is much lower than on the previous system (fonts nearly unreadable, jagged lines etc.).

The monitor is the same as previously (a Siemens device). The Windows Media Player has been upgraded to Version 9 (9.00.00.2980), however the same version of the player on the old machine with a far worse monitor gives better results!

After some experimentation, it was found that the problem is with the refresh rate setting of the monitor. This value has been set as high as possible (90 Hz at 1600x1200 resolution) to avoid flickering which may disturb editing work. However, the performance in animation playing is disastrous. The only acceptable frequency for animation playing is as low as 60 Hz (which gives some flickering, and is therefore inacceptable for normal work).

The following work-around solution is adopted:

- Keep monitor frequency at 90 Hz (1600x1200, 16-bit color) for normal work
- When displaying animations, turn temporarily the frequency to 60 Hz (use the icon in the lower right part of the status bar on the desktop).

10.7.4 UnxUtils

The new version of the Unix commands (see Section 2.4.3) has been obtained and installed. This consists of the two packages `UnxUtils.zip` and `UnxUpdates.zip`. Unpack them by hand, and replace all executable files into `D:\Users\Folco\Run\Gnu`. Do not forget also the shell executable (`sh.exe`) which is located on a separate sub-directory in the distribution. This command may be handy for typing on-the-fly ad-hoc scripts from the keyboard.

Note that a single command `type.exe` may create problems because of the name clash with DOS's `type` command. To avoid this problem, move this executable to `D:\Users\Folco \Run\Gnu_inactive`.

10.7.5 Netpbm

The new version of the Netpbm utilities has been downloaded. These are heavily used, among other things, for the production of reports and in particular of animations. The current version id 10.6-1. The packages are `netpbm-10.6-1-lib.zip`, `bin.zip`, `doc.zip` and (optionally) `src.zip`. They are installed by unpacking under `C:\Netpbm`. Strangely, the current version lacks the PostScript and PDF version of the man pages. The ones from the previous installation have been retained (files `ps\netpbm-man.ps` and `pdf\netpbm-man.pdf`).

10.7.6 Perl

The new version 5.8.0 (Build 804) automatically performs (like the previous one) file association when installed, thus the manual setting described in Section 3.1.1 is no longer necessary. However, the `PATHEXT` variable setting (Section 3.1.2) is still required.

The behaviour concerning the globbing (expansion of file names) is the same as in the previously installed version (see Section 10.5.1), so no additional modification is required in the procedures.

The Sendmail package (`Sendmail.pm`) has been installed in `C:\Perl\Lib\Mail` and the date package (file `C:\perl\site\lib\HTTP\Date.pm`) has been customized according to the instructions in Section 3.6.

A nasty problem has been encountered with the new version of Perl in procedure `evototgz.pl`, described in Section 6.15. After some experimentation, the problem was found to concern the instruction:

```
open (INPUT, "type $evo |");
while (<INPUT>) {
   . . .
}
```

The file `$evo` was opened, but after reading the first line the while loop terminated, although there were other lines in the input file.

The strange thing is that several other procedures used the same syntax, without problems. Also, building up a simplified example with the above lines of Perl code did work correctly. The problem may be due to the more complex actions that are performed within the `while` loop in the `evototgz` procedure, but it was not possible to detect the real reason.

It has been decided to replace the above lines with the alternative form:

```
open (INPUT, $evo);
while (<INPUT>) {
   . . .
}
```

which is perfectly equivalent, and also simpler. This eliminates the problem with `evototgz.pl`. All other utility procedures that contained the same type of code have been updated to the simpler syntax.

Head command

Another problem encountered is the fact that this version of Perl contains a command `HEAD.bat` in `C:\Perl\Bin`. The name clashes with the standard Unix command head, that is part of the Gnu utilities (See Section 10.7.4). Since the Perl directory is found on the `PATH` before the directory hosting the Gnu commands, the Perl version is executed. This is a problem because several of the other procedures need the Unix version of `head`. Since this command is also very useful from the command line, it is decide to disactivate the Perl command by renaming the file `HEAD_ori.bat`.

10.7.7 Vim

The new version 6.1 of the Vim editor needs the same customization as the previous 6.0 version, described in Section 10.5.2. The configuration file `_vimrc` has been copied from the previous installation, because the base file `C:\Vim\Vim61\vimrc_example.vim` has not changed since.

The same applies to syntax files `eiffel.vim` and `fortran.vim`. However, `filetype.vim` has changed slightly, so the customization described in Section 10.5.2 has been re-applied to the newer version.

The package installs in `C:\Vim`.

10.7.8 MiKTeX

As already described in Section 10.5.3 for the previous installation (Version 2.1), the current MiKTeX package (Version 2.2) uses an installer `setup.exe` that is used to download, install and continuously update the various packages that compose the installation. A so-called "large" system installation was chosen (about 100 MB on disk).

After installation, it is necessary to perform the file association of `.dvi` files with the `yap` previewer (`C:\texmf\miktex\bin\yap.exe`) by hand (***Tools → Folder options***). Do it as administrator to have it automatically also for the other users.

A small malfunctioning has been observed in the current version of `yap`. When started, it sometimes complains that fonts are not available. Usually, by closing the program and starting it up again the problem disappears (maybe it builds up the fonts on-the-spot, but why the error message then?).

The same customizations described in Sections 7.5.9 and 7.5.10 have been re-applied. The old versions of the non-standard LaTeX packages (see Table in Section 7.5.9) have been re-used (except `hevea.sty`, see below). However, the customization of file `miktex.ini` has been re-applied on the newer version of this file, which had slightly changed since the previous installation.

10.7.9 Hevea

The `hevea` package is now at version 1.06, at least in the Windows port, which is the one needed here. Installation was performed as described in Section 10.5.4 for the previous version.

10.7.10 Tth

The `tth` package is no longer contained in the standard MiKTeX distribution. Use the old package. Installation is done as explained in Section 10.5.5.

10.7.11 Ghostscript and GSview

The current versions are Ghostscript 8.00 (single self-extracting file `gs800w32.exe`) and GSView 4.3 (single self-extracting file `gsv43w32.exe`). Installation consists simply in double-clicking the self-extracting executable files and following the instructions.

By default, Ghostscript installs in `C:\Gs` while GSview (unlike the previous version!) installs in `C:\Program Files\Ghostgum`.

10.7.12 PStoEdit

Note that the distribution of GSview no longer (unlike the previous version!) includes PstoEdit, therefore this package has been obtained separately (`pstoeditsetup.exe`). The two libraries `msvcr70.dll` and `msvcp70.dll` are also necessary. They may be downloaded from a link in the PstoEdit page. However, MiKTeX installs these two files (same date and size) under `C:\texmf\miktex\bin` (which is in the `PATH`), so they were not copied to the pstoedit folder as suggested.

While installing, remember to uncheck installation of the (optional) ImageMagick add-on, else when started it complains that ImageMagick is not present on the system.

To verify PstoEdit, open a document with GSview, the click on ***Edit → Convert to vector format***. Many vector formats, including for example Windows Metafile (WMF) should be available. This allows to convert any PostScript image to Windows Metafile for inclusion, e.g., in a Word document.

10.7.13 Installation of Adobe products

A series of (commercial) products from Adobe, whose installation and set-up is quite tricky, is heavily used at our site. Although these products are not part of the mirror site proper, some guidelines are nevertheless given for their correct installation and tuning.

Initialization

Before installing any Adobe products, it is best to uninstall any previously installed products of the same type, and closely follow the instructions below, in the given order. All installations should be performed under the administrator account.

Adobe Type Manager

This product is (unlike in the previous installation) no longer contained in the FrameMaker package (now at version 7.0). A "free" version of the tool (Type Manager Light) may be obtained from the Adobe Web site, however it seems that this product is no longer needed (and probably it was already so under Windows 2000, but not for NT for example). So it was decided not to install it.

FrameMaker 7.0

Install a default PostScript printer on your system, if not already available. For example, the hp4000_ps printer. Choose it as your default printer.

Then, install FrameMaker 7.0. Insert the FrameMaker 7.0 distribution CD and choose to install FrameMeker, with International English interface and Registered Owner Version, with all dictionaries. Once terminated, the installer proposes to install also Distiller. Reply no to this question (Distiller will be installed separately with Acrobat 5.0, see below). The same applies for the AdobePS PostScript driver and for Acrobat Reader. The installed version of FrameMaker is 7.0p492.

Acrobat 5.0

Next, install Acrobat 5.0 from the separate CD (not the version that comes on the FrameMaker 7.0 CD) plus the Distiller. Follow the instructions, the installation is smooth. The installed version is 5.05 for

Acrobat and 5.0.5.2001101100 for the Distiller. The installation directory is `C:\Program Files\Adobe\Acrobat5.0`.

It is important to note that this installation will produce two extra "printers" in your control panel settings, namely:

- Acrobat Distiller
- Acrobat PDFWriter

Neither of these will be actually used, but leave them for completeness (they are harmless).

Adobe PostScript Driver

The installation of this component is somewhat tricky. Unfortunately, it seeem absolutely necessary to install it, else the PostScript files produced by FrameMaker are often incompatible with both GSview and the distiller.

The installer, available from Adobe, is the executable file `WINSTENG.EXE`. It contains versions for Windows 98, Nt, 2000 and XP and automatically recognizes the correct system upon installation. Under XP, it should install a driver called Pscript5 5.2.

Before installing, it is necessary to get a PPD (PostScript Printer Description) file for the "printer" that will be used. In our case, we do not want a real printer, but a generic PostScript printer to print from FrameMaker to a file. This file will then be distilled by Acrobat to obtain the PDF version. The necessary PPD is therefore the one corresponding to the Acrobat Distiller printer. The required PPD file was downloaded from the Adobe site, and is called `ADIST5.PPD`.

Copy this file to a temporary place (for safety), then launch `WINSTENG.EXE`:

- Choose: Install a New PostScript printer;
- How is your Printer connected: choose Directly (Local printer);
- Local port selection: choose FILE;
- Select printer model: the only available one in the list is Generic PostScript Printer, that would install a file `DEFPRTR2. PPD`. This is wrong!. Click on Browse and locate the `ADIST5.PPD` (Acrobat Distiller) file instead;
- Choose Not Shared;
- Under Printer Name type: Generic PS Printer (distiller);
- Should this be your Default Printer: answer No;
- Should test page be printed: answer No;
- Configure: answer Yes, make sure the only available paper size is A4.

To test out your freshly installed driver, first make sure that the new "printer" named 'Generic PS Printer (distiller)' is available in your printers list. Then, open a FrameMaker document and try printing to this printer (actually, to a PostScript file through this driver). Visualize and check the PostScript file by GSview. Convert it to PDF by Acrobat Distiller.

The nasty points where all other drivers, more or less, seem to fail are the following:

- Mathematical formulas containing characters with a tilde under them (to represent a vector, matrix or tensor): the tildes tend to disappear!
- Some large symbols (summations for example) have a tendency to disappear.

- Verify the correct treatment of colors, they should be visible both in GSview and in the PDF file.

Note that when you choose to print to the new driver from FrameMaker, it may warn that some fonts may be substituted. This is usually not a major problem.

Settings for PDF output

It has been noted that in some cases the size of the final resulting PDF files is much larger (3-4 times) the one obtained with the previous installation. Also the PostScript file is larger, about twice the previous size.

These problems may be avoided by adequate settings in both FrameMaker 7.0 and Distiller 5.0. These are mandatory especially when opening a document produced with a previous version of FrameMaker. Do the following. First, produce a customized configuration of the Distiller (note that, strangely, this was not necessary with the previous version of the software):

- Launch the distiller and choose ***Job Options: Screen***. Then click on ***Settings → Job Options → Compression*** and uncheck all three boxes related to image downsampling (Color, Grayscale and Monochrome images). Leave the compression settings unchanged. Then click on "Save As" and save this configuration as Folco.

- Remember to always use this Folco configuration when distilling. This customization is necessary, else the results for bitmap images (e.g. the ones obtained via pstogif) are extremely poor. The setting of the Folco configuration may be done also in Word (see under ***Acrobat → Change Conversion Settings → Settings → General***: choose the Folco configuration) and in FrameMaker (see below).

Every "old" FrameMaker document or book must be "optimized" for the new PDF version:

- Open the document in FM 7.0. If it is a book, open all files. Choose ***Format → Document → Optimize PDF Size → Options***. Set ***Optimize Size of All Linked Documents*** and check the boxes ***Force Optimization*** and ***Clear Existing Optimization Info***. Uncheck the ***Prompt …*** boxes and check ***Overwrite Existing Files*** and ***Cancel on Error***.

- Choose ***Format → Document → Optimize PDF Size → Optimize File***, select the document (or book) and click ***Select***. The document or book is optimized.

Next, when you want to transform a FM document or book to PDF, open it and then:

- Select all files in book and choose ***Format → Document → PDF Setup***. Or, perhaps more simply, click on ***File → Print Book → Setup*** and choose the Generic PS Printer (distiller).

- Click on ***Properties***, in the ***Paper/Quality*** Tab make sure ***Color*** is checked. Click on ***Advanced***, make sure ***Paper Size*** is A4, and ***Graphic → Print quality*** is 600 dpi (not 1200).

- Then activate ***Print to File*** and ***Generate Acrobat Data*** and click on ***PDF Setup***. In either way the successive PDF setup refers to all files in the book.

- On the ***Settings*** Tab, choose ***PDF Job Options: Folco***, uncheck ***Generate Separate PDF …***, set ***Page Range: All***. On the ***Bookmarks*** Tab activate ***Generate PDF Bookmarks*** and choose the desired paragraphs to include.

- On the ***Tags*** Tab, uncheck ***Generate Tagged PDF*** (to reduce file size).

- On the ***Links*** Tab, uncheck ***Create Named Destinations for All Paragraphs***, to reduce file size.

- Print document to PS file.

- Now you may distill the PS file. Make sure that options in Distiller are as follows. The ***Job Options*** setting must be to Folco, ***Compatibility*** should be Acrobat 3.0 (PDF 1.2).

In this way, the size of both the PostScript file and the final PDF file should be relatively small (like in the previous installation), without too much compromising the quality of results.

Illustrator

The previous version of Illustrator is not compatible with XP. A new product and license will be obtained and installed as soon as possible.

Fixing Adobe program group

The described installation produces separate program start entries in the Start menu: Adobe (containing only FM 7.0), Adobe Acrobat 5.0 and Adobe Distiller 5.0. These may be regrouped under the Adobe folder.

To fix this (as administrator), you must cut the `Acrobat Distiller 5.0` and the `Adobe Acrobat 5.0` folder from `C:\Documents and Settings\AllUsers\Start Menu\Programs` and paste them under `C:\Documents and Settings\All Users\Start Menu\Programs\Adobe`.

You may act similarly also to group all other Adobe applications (e.g., Illustrator) under the same program group.

Fixing file associations

After all these installations, file associations are probably not what one would like to have. This may be fixed (under administrator) and is automatically available to all users. The adjustments are done in Windows Explorer by clicking on ***Tools → Folder Options → File Types***, and by selecting the following file extensions: .PS (for PostScript files, note that this applies automatically also to .EPS, i.e. encapsulated PostScript files), and .PDF.

The suggested settings are:

- .PS file default action: `open`. Application: `"C:\Program Files\Ghostgum\gsview\gsview32.exe" "%1"`. No DDE.
- .PS file action `print`. Application: `"C:\Program Files\Ghostgum\gsview\gsview32.exe" /p "%1"`. No DDE.
- .PS file action `PDF`. Application `"C:\Program Files\Adobe\Acrobat 5.0\Distillr\AcroDist.exe" %1`. No DDE.
- .PDF file default action: `open`. Application: `"C:\Program Files\Adobe\Acrobat 5.0\Acrobat\Acrobat.exe" %1`. No DDE.
- .PDF file action: `open with gsview`. Application: `"C:\Program Files\Ghostgum\gsview\gsview32.exe" "%1"`. No DDE.

10.8 Security issues

At the end of 2002 security issues were raised both at CEA (hosting the central site) and at JRC (hosting a mirror site) as far as communication over the Internet is concerned. Some accesses via `ftp` or Web were restricted. To cope with these changes, the following actions were taken.

10.8.1 Telnet and Ftp with the Saclay machine

The `telnet` and `ftp` commands from JRC to acces the CEA machine do not work any more. However, similar functionality may now be obtained by using secure (`SSH`) protocol via the following commands:

- `putty` (a replacement for `telnet`). Make sure to check the `SSH` button, then login as usual.

- `psftp` (a replacement for `ftp`).
- `pscp`

The Windows executables of these three tools may be downloaded from the following internet address: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html.

10.8.2 Transfer of evolution packages

Direct transfer of evolution packages from the central site to the JRC mirror site is no longer possible, because the JRC site is now hosted on a machine (`SM48`) that resides inside the JRC firewall, for security reasons.

To circumvent this problem, the transfer is performed in two steps:

- First, the central site transfers the files to a machine which is publicly accessible (outside the JRC firewall). This machine is named `europlexus.jrc.it` and is currenly a Linux machine. Appropriate username and password are required to `ftp` files to this machine. The files are then automatically placed in a pre-determined directory.
- Next, the files are moved from the "public" machine to the JRC mirror site (Windows machine inside the firewall). This operation is executed by the Windows machine, which executes a new procedure `epx_ftp_getfiles`. This is a Perl procedure using the `ftp` package `NET::FTP` to access the Linux machine.
- Finally, the evolution may start as usual, by using the files on the mirror site.

The syntax of the new command is:

```
epx_ftp_getfiles
```

and is listed in the Appendix. It accesses the Linux machine by `ftp`, copies any evolution files from the pre-defined directory to the mirror site and finally removes these files from the Linux machine. Any files whose names do not match an evolution file name are ignored.

The `epx_evol_start` procedure has been modified accordingly. Now, at the very beginning of this procedure, the `epx_ftp_getfiles` command is executed to transfer any evolution files before starting the evolution proper.

10.9 Consortium Web Site

A prototype of Web site for the EUROPLEXUS Consortium is being set up at JRC. This site is located on the Linux machine outside the JRC firewall, and is intended to gradually fulfil a twofold purpose:

- Offer information about the EUROPLEXUS Consortium and its activities. This part is freely accessible by anybody on the Web. It will contain advertising material and publicly available documents, brochures, demonstrative animations etc. At the moment, it contains just a link to the HTML version of the User's Manual A link to the commercial EUROPLEXUS site (hosted by Samtech) will also be provided, as soon as this will be available.
- Act as a repository and forum for exchange of information among the partners of the EUROPLEXUS Consortium. This part of the site is restricted and accessible only via a user name and password identifying the authorized partner. At the moment, it offers the following services: access to the User's Manual (both HTML and PDF versions), to the current executable module for Windows, to the list of available on-line bibliography (including also some data files and animations) and a link to the previous PLEXIS-3C Web page (this is now partly obsolete, but it still contains some valuable examples and animations).

The manuals and executable are continuously updated at each evolution of the JRC mirror site. The update occurs via `ftp` from the mirror site machine towards the Linux machine, via two new procedures:

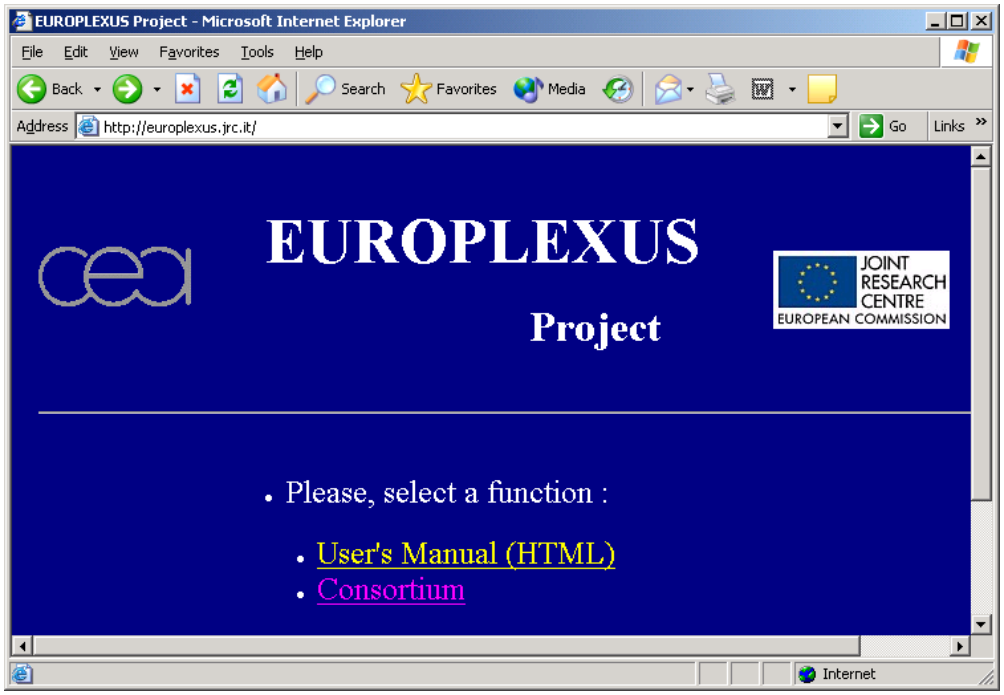
```
epx_ftp_putexe [-w] [-M] [-c]
```

updates the executable (the standard version by default, or the QuickWin version by using the `-w` switch), or the MPI version using the `-M` switch, or the –check version using the `-c` switch, and

```
epx_ftp_putman
```

updates the manuals. These are listed in the Appendix. The `epx_evol_start` procedure has also been updated and now calls the new procedures automatically.

The EUROPLEXUS Development Center currenly hosted at Saclay has been updated accordingly. Now the main Web page contains a link “Consortium” that accesses the Consortium Web page. The address of this page is http://europlexus.jrc.it and it looks as follows:



By clicking on “User’s Manual (HTML)” the freely accessible version of the manual appears. By clicking on “Consortium” and by entering the required user name and password, the reserved part of the site becomes available, see below.

From this page, all the services listed below are accessible. Additional services are planned to be introduced shortly.

10.10 OpenGL based graphical module

In the first months of 2003 the development of a new built-in graphical module for interactive and non-interactive post-processing of EUROPLEXUS results has been started, see references [12, 13, 14]. The module relies upon the OpenGL graphical language which is a de-facto standard for advanced 3D rendering, and is freely available on virtually any platform.

This module is meant to gradually replace the somewhat outdated graphical package based upon QuickWin (see [6]), with the following advantages:

- Graphical representation is far superior and much more efficient, since OpenGL exploits the power of modern graphic cards, while in the previous implementation rendering was very limited and done entirely in software.

- The program may now be compiled as a console application and thus the difficulties and inefficiencies related to the use of QuickWin, see e.g. Section 10.1.1, are completely avoided.

However, for a transition period, it has been decided to keep two versions of the executable. The first (standard) one is built as a console application, with the possibility of OpenGL-based rendering but with QuickWin graphics disabled. The second one is built as a QuickWin application and contains both types of graphics. However, be aware that due to apparent incompatibilities (to be further investigated) between QuickWin and OpenGL, if one attempts to use both types of graphics in the same execution of the program, the code may sometimes block (typically under debugger, when an OpenGL window is closed and control should return to the QuickWin part). Therefore, the QuickWin executable is meant to be used exclusively for QuickWin graphics.

10.10.1 Additional software and libraries

To implement the OpenGL part, use is made of the following software:

- **OpenGL**, is the core graphical package which may be obtained freely and usually comes bundled with the operating system, at least on (recent) MS-Windows machines (from the NT 4 operating system onwards).

- **GLU**, the OpenGL Utility library, is also widely available and is usually bundled with the operating system.

- **GLUT**, the OpenGL Utility Toolkit library, written by Mark Kilgard, may be obtained freely, but necessitates some minor modifications in order to be used within EUROPLEXUS, as described in reference [12].

- **F90GL**, a Fortran 90 binding of OpenGL (and GLU/GLUT) functions, which are written in C. This package, written by William Mitchell, allows to program the EUROPLEXUS rendering modules directly in Fortran 90, which is the standard programming language used in the code. This package also necessitates some modifications with respect to the standard version (see previous point and reference [12]).

- **Bmplib**, a small library that provides functions to manipulate (read/write) bitmap files (`.BMP`) from within OpenGL. This functionality is missing in the standard GLUT, and has been added in view of the large importance it has for producing images for publications and for creating animation files. The library is described in reference [13].

As a consequence of these additions, some extra files and libraries with respect to the previous situation are needed for the development and use of EUROPLEXUS. The files are either C header files (`.H`) or Fortran 90 object module files (`.MOD`), and are used at compilation time. The libraries are of two types, namely static libraries (`.LIB`), which are necessary during the link phase in order to generate an executable module, and dynamically linked libraries (`.DLL`), which are loaded at run time. These are described in detail in references [12, 13, 14] and are also summarized hereafter for convenience:

- `glut32.dll`, is the GLUT dynamic link library, and should be copied to an accessible directory, normally `%WINDIR%\SYSTEM32`, where also other DLLs reside. On the site machine, this directory is `C:\WINDOWS\SYSTEM32`. This library is necessary in order to run the executable.

- `glut32.lib`, is the GLUT static library, and should be copied to an accessible directory, normally `$(MSDevDir)\..\..\VC98\Lib`, where also other libraries used at link time reside. On the site machine, this directory is `C:\Program Files\Microsoft Developer Studio\VC98\Lib`. This library is necessary in order to link the executable.

- `glut.h`, is the GLUT C header file, and is only necessary to generate the modified GLUT package itself. It should be copied to an accessible directory, normally `$(MSDevDir)\..\..\VC98\Include\GL`, where also other headers used at compilation time reside. On the site machine, this directory is `C:\Program Files\Microsoft Developer Studio\VC98\Include\GL`.

- `Glutf90.h`, is the GLUT Fortran 90 binding C header file, and is only necessary to generate the modified F90GL package. See previous point.

- `f90GL.lib`, is the OpenGL Fortran 90 binding static library, and should be copied to an accessible directory, normally `$(MSDevDir)\..\..\DF98\Lib`, where also other Fortran libraries used at link time reside. On the site machine, this directory is `C:\Program Files\Microsoft Developer Studio\DF98\Lib`. This library is necessary in order to link the executable.

- `f90GLU.lib`, is the GLU Fortran 90 binding static library, see previous point.

- `f90GLUT.lib`, is the GLUT Fortran 90 binding static library, see previous point.

- opengl_fwrap.mod, is a Fortran 90 binary module file needed for the generation and use of the F90GL package. It should be copied to an accessible directory, normally $(MSDevDir)\..\..\DF98\Include, where also other Fortran object modules reside. On the site machine, this directory is C:\Program Files\Microsoft Developer Studio\DF98\Include.

- opengl_gl.mod, see previous point. Needed also when compiling and linking the code.

- opengl_glinterfaces.mod, see previous point.

- opengl_glu.mod, see previous point. Needed also when compiling and linking the code.

- opengl_gluinterfaces.mod, see previous point.

- opengl_glut.mod, see previous point. Needed also when compiling and linking the code.

- opengl_glutinterfaces.mod, see previous point.

- opengl_kinds.mod, see previous point.

- bmplib.lib, is the bmplib static library, used at link time to provide access to the bitmap manipulation functions. This library is written in C and access from Fortran 90 is achieved by appropriate INTERFACE statements that are inserted directly in the EUROPLEXUS modules [13] (unlike in the case of GLUT, which uses a separate package F90GL). It should be copied to an accessible directory, normally $(MSDevDir)\..\..\DF98\Lib, where also other Fortran libraries used at link time reside. On the site machine, this directory is C:\Program Files\Microsoft Developer Studio\DF98\Lib. This library is necessary in order to link the executable.

- gdi32.lib, is a Windows system library containing GDI functions used by bmplib (see [13]). On the site machine, this library is located in C:\Program Files\Microsoft Developer Studio\VC98\Lib . The library may or may not be automatically included depending upon the project type. Make sure that it is always available and include it explicitly in the link command if needed.

10.10.2 Implications on standard commands

Some of the standard code development commands described in the previous Sections have been updated, in order to account for the changes related to the new Graphics package. In particular:

- epx_filbat.bat (see Section 8.1) now uses the two standard filter keywords WIN OGL instead of WIN QWIN.

- epx_filbat_quickwin.bat (see Section 8.1) is a new file, similar to the previous one but using the three keywords WIN OGL QWIN.

- epx_cmp (see Section 5.4) now uses the two standard filter keywords WIN OGL instead of WIN QWIN. The syntax of the command is unchanged.

- epx_lk (see Section 5.5) now builds by default a console application (like before) but with the OpenGL graphics enabled (the QuickWin graphics is disabled and an error message occurs if the user types trac … norm). Optionally (like before) the user may request to build a QuickWin application (by epx_lk -w …). The syntax of the command is therefore unchanged. In this case, the script retrieves the two sources ifwin.ff and m_qwin.ff, which are the only ones containing QuickWin-related statements, compiles them by providing

the additional keyword `QWIN`, and links as a QuickWin application. The resulting executable is meant for use with QuickWin graphics and is now named by default `epxqw.exe`. The OpenGL graphics is not disabled, but its functionality may not be guaranteed because of compatibility problems with QuickWin that have been reported on some machines.

- `epx_mkbatch` (see Section 6.5.1) is now obsolete and should no longer be used. This is because the standard executable is now a console application and may be used directly in batch mode (possibly by `epx_bench -b` to suppress most terminal output).

- `epx_bench` (see Section 5.6) has a new switch `-w` by which it will use the current standard QuickWin executable (`%EUROPLEXUS%\Exe\europlexusqw.exe`) instead of the current standard Console executable.

- `epx_test_benchmarks` (see Section 6.5) no longer needs the `-b` switch which was previously used to run the batch (`epxb.exe`) executable instead of the QuickWin one. Therefore, this switch has been removed from the command. By default, the command runs the local executable (`epx.exe` by default), which now should be a batch executable with OpenGL graphical capabilities. To use a QuickWin executable (`epxqw.exe` by default) use the `-e` switch, i.e. the command: `epx_test_benchmarks -e epxqw.exe`.

- `epx_init` (see Section 8.1) now produces a local project of Console application type instead of QuickWin application type. To debug the QuickWin version, use `epx_init -w` (see Section 8.3).

- `epx_evol_start` (see Section 6.3) now produces two standard executables, the Console version (`europlexus.exe`) and the QuickWin version (`europlexusqw.exe`). However, all tests are performed by the console version and the other one is used just to update that version both in the `exe` directory and on-line.

- `epx_make` (see Section 6.13) has been updated to use `epx_ordo` to find the correct compilation order. Furthermore, it uses the standard executable to run the benchmarks instead of building up an ad-hoc batch version.

10.11 Version March 2004

On March 2004 a complete re-installation of the mirror site was performed, due to hardware failure of the hard disk of the PC hosting the site (sm48). All the public-domain tools were updated to the most recent available versions. Here is a short list of the settings that have been necessary.

10.11.1 Operating system

The version of the operating system is still MS Windows XP Professional, Service Pack 1. The machine is a Pentium 4 at 2.0 GHz with 512 MB of RAM.

10.11.2 Developer Studio and Compilers

The installed compilers and programming environment are the same as under the previous installation:

- Microsoft Visual Studio 6.0 Professional Edition, with Service Pack 5.

- Visual C++ 6.0 Compiler

- Compaq (now Intel) Visual Fortran Professional Edition 6.6

As concerns the Fortran compiler, the installation proceeds in two steps: first install Version 6.5, then install the upgrade to 6.6.

The same patch as described in Section 10.2.2 to the Fortran compiler has been installed to avoid the problem with the `BACKSPACE` instruction.

10.11.3 Animations quality problem

This problem (see Section 10.7.3) persists because the hardware has not been changed.

10.11.4 WinZip

The evaluation version 9.0 of the WinZip package has been downloaded from the network and installed. This consists of the single file winzip90.exe (click to install).

10.11.5 UnxUtils

This package has not been updated and we are still using the previous version.

10.11.6 Netpbm

This package has not been updated and we are still using the previous version (10.6-1).

10.11.7 Perl

The new version 5.8.3 (Build 809) consists of the single self-installing file ActivePerl-5.8.3.809-MSWin32-x86.msi, which automatically performs (like the previous one) file association when installed, thus the manual setting described in Section 3.1.1 is no longer necessary. However, the `PATHEXT` variable setting (Section 3.1.2) is still required.

The behaviour concerning the globbing (expansion of file names) is the same as in the previously installed version (see Section 10.5.1), so no additional modification is required in the procedures.

The Sendmail package (`Sendmail.pm`) has been installed in `C:\Perl\Lib\Mail` and the date package (file `C:\perl\site\lib\HTTP\Date.pm`) has been customized according to the instructions in Section 3.6.

Head command

The same problem encountered in the previous installed version, namely the fact that this version of Perl contains a command `HEAD.bat` in `C:\Perl\Bin`, is still encountered here. The name clashes with the standard Unix command head, that is part of the Gnu utilities (See Section 10.7.4). Since the Perl directory is found on the `PATH` before the directory hosting the Gnu commands, the Perl version is executed. This is a problem because several of the other procedures need the Unix version of `head`. Since this command is also very useful from the command line, it is decide to disactivate the Perl command by renaming the file `HEAD_ori.bat`.

10.11.8 Vim

The new version 6.2 of the Vim editor needs the same customization as the previous 6.1 version, described in Sections 10.5.2 and 10.7.7. The configuration file `_vimrc` has been copied from the previous installation, because the base file `C:\Vim\Vim62\vimrc_example.vim` has changed only slightly since the previous version.

The same applies to syntax file `eiffel.vim`. Note that, contrary to what had been stated for the previous customization, the file `fortran.vim` does not need any changes (it was already so in version 6.0). However, `filetype.vim` has changed slightly, so the customization described in Section 10.5.2 has been re-applied to the newer version.

The package installs in `C:\Vim`.

10.11.9 Ghostscript and GSview

The current versions are Ghostscript 8.14 (single self-extracting file `gs814w32.exe`) and GSView 4.6 (single self-extracting file `gsv46w32.exe`). Installation consists simply in double-clicking the self-extracting executable files and following the instructions.

By default, Ghostscript installs in `C:\Gs` while GSview (unlike the previous version!) installs in `C:\Program Files\Ghostgum`.

10.11.10 PStoEdit

It was decided to not install PstoEdit any more.

10.11.11 MiKTeX

The current MiKTeX package (Version 2.4) uses an installer `setup.exe` that is used to download, install and continuously update the various packages that compose the installation. A so-called "large" system installation was chosen (about 100 MB on disk).

After installation, it is no longer necessary to perform the file association of `.dvi` files with the `yap` previewer, unlike with the previous version 2.2.

The same customizations described in Sections 7.5.9 and 7.5.10 have been re-applied. The old versions of the non-standard LaTeX packages (see Table in Section 7.5.9) have been re-used (except `hevea.sty`, see below). However, the customization of file `miktex.ini` has been re-applied on the newer version of this file, which had slightly changed since the previous installation.

10.11.12 Hevea

An attempt was made to use the new version of the `hevea` package (1.07). This worked, except for the fact that the index was not built and the file `epx_pass_4.log` contained lots of error messages concerning unrecognized `\index` commands. Therefore, the previous version 1.06 was reused. Installation was performed as described in Section 10.5.4 for the previous version.

10.11.13 Tth

The `tth` package is no longer contained in the standard MiKTeX distribution. Use the old package. Installation is done as explained in Section 10.5.5.

10.11.14 Installation of Adobe products

A series of (commercial) products from Adobe, whose installation and set-up is quite tricky, is heavily used at our site. Although these products are not part of the mirror site proper, some guidelines are nevertheless given for their correct installation and tuning.

Initialization

Before installing any Adobe products, it is best to uninstall any previously installed products of the same type, and closely follow the instructions below, in the given order. All installations should be performed under the administrator account.

FrameMaker 7.0

Install a default PostScript printer on your system, if not already available. For example, the hp4000_ps printer. Choose it as your default printer.

Then, install FrameMaker 7.0. Insert the FrameMaker 7.0 distribution CD and choose to install FrameMaker, with International English interface and Registered Owner Version, with all dictionaries. Once terminated, the installer proposes to install also Distiller. Reply no to this question (Distiller will

be installed separately with Acrobat 6.0, see below). The same applies for the AdobePS PostScript driver and for Acrobat Reader. The installed version of FrameMaker is 7.0p492.

Acrobat 6.0

Next, install Acrobat 6.0 pro from the separate CD (not the version that comes on the FrameMaker 7.0 CD) plus the Distiller. Follow the instructions, the installation is smooth. The installed version is 6.0.0 for Acrobat and 6.0.0 for the Distiller.

It is important to note that this installation will produce an extra "printer" in your Start -> "Printers and Faxes" settings, namely:

- Adobe PDF (Adobe PDF Converter)

In Start -> "Printers and Faxes" select this printer and choose Printing Preferences by clicking the right mouse button.Under Layout -> Advanced (or Paper/Quality -> Advanced) change paper size from Letter to A4 and Graphic Print Quality from 1200 dpi to 600 dpi, and click OK IN Adobe PDF Settings change Adobe PDF Page Size to A4 and click Apply.

Adobe PostScript Driver

The installation of this component is somewhat tricky. Unfortunately, it seeem absolutely necessary to install it, else the PostScript files produced by FrameMaker are often incompatible with both GSview and the distiller.

The installer, available from Adobe, is the executable file `WINSTENG.EXE`. It contains versions for Windows 98, Nt, 2000 and XP and automatically recognizes the correct system upon installation. Under XP, it should install a driver called Pscript5 5.2.

Before installing, it is necessary to get a PPD (PostScript Printer Description) file for the "printer" that will be used. In our case, we do not want a real printer, but a generic PostScript printer to print from FrameMaker to a file. This file will then be distilled by Acrobat to obtain the PDF version. The necessary PPD is therefore the one corresponding to the Acrobat Distiller printer. The required PPD file was downloaded from the Adobe site, and is called `ADIST5.PPD`.

Copy this file to a temporary place (for safety), then launch `WINSTENG.EXE`:

- Choose: Install a New PostScript printer;
- How is your Printer connected: choose Directly (Local printer);
- Local port selection: choose FILE;
- Select printer model: the only available one in the list is Generic PostScript Printer, that would install a file `DEFPRTR2.PPD`. This is wrong!. Click on Browse and locate the `ADIST5.PPD` (Acrobat Distiller) file instead;
- Choose Not Shared;
- Under Printer Name type: Generic PS Printer (distiller);
- Should this be your Default Printer: answer No;
- Should test page be printed: answer No;
- Configure: answer Yes, make sure the only available paper size is A4.

To test out your freshly installed driver, first make sure that the new "printer" named 'Generic PS Printer (distiller)' is available in your printers list. Then, open a FrameMaker document and try printing to this printer (actually, to a PostScript file through this driver). Visualize and check the PostScript file by GSview. Convert it to PDF by Acrobat Distiller.

The nasty points where all other drivers, more or less, seem to fail are the following:

- Mathematical formulas containing characters with a tilde under them (to represent a vector, matrix or tensor): the tildes tend to disappear!
- Some large symbols (summations for example) have a tendency to disappear.
- Verify the correct treatment of colors, they should be visible both in GSview and in the PDF file.

Note that when you choose to print to the new driver from FrameMaker, it may warn that some fonts may be substituted. This is usually not a major problem.

Settings for PDF output

It has been noted that in some cases the size of the final resulting PDF files is much larger (3-4 times) the one obtained with the previous installation. Also the PostScript file is larger, about twice the previous size.

These problems may be avoided by adequate settings in both FrameMaker 7.0 and Distiller 6.0. These are mandatory especially when opening a document produced with a previous version of FrameMaker. Do the following. First, produce a customized configuration of the Distiller (note that, strangely, this was not necessary with the previous version of the software):

- Launch the distiller and choose ***Default Dettings: Standard***. Then click on ***Settings → Edit Adobe PDF Settings → General*** and set Default page width to 21 cm and default page height to 29.7 cm. Then click on ***Images*** and uncheck all three boxes related to image downsampling (Color, Grayscale and Monochrome images). Leave the compression settings unchanged. Then click on "Save As" and save this configuration as Folco.
- Remember to always use this Folco configuration when distilling. This customization is necessary, else the results for bitmap images (e.g. the ones obtained via pstogif) are extremely poor. The setting of the Folco configuration may be done also in Word (see under ***Acrobat → Change Conversion Settings → Settings → General***: choose the Folco configuration) and in FrameMaker (see below).

Every "old" FrameMaker document or book must be "optimized" for the new PDF version:

- Open the document in FM 7.0. If it is a book, open all files. Choose ***Format → Document → Optimize PDF Size → Options***. Set ***Optimize Size of All Linked Documents*** and check the boxes ***Force Optimization*** and ***Clear Existing Optimization Info***. Uncheck the ***Prompt …*** boxes and check ***Overwrite Existing Files*** and ***Cancel on Error***.
- Choose ***Format → Document → Optimize PDF Size → Optimize File***, select the document (or book) and click ***Select***. The document or book is optimized.

Next, when you want to transform a FM document or book to PDF, open it and then:

- Select all files in book and choose ***Format → Document → PDF Setup***. Or, perhaps more simply, click on ***File → Print Book → Setup*** and choose the Generic PS Printer (distiller).
- Click on ***Properties***, in the ***Paper/Quality*** Tab make sure ***Color*** is checked. Click on ***Advanced***, make sure ***Paper Size*** is A4, and ***Graphic → Print quality*** is 600 dpi (not 1200).
- Then activate ***Print to File*** and ***Generate Acrobat Data*** and click on ***PDF Setup***. In either way the successive PDF setup refers to all files in the book.
- On the ***Settings*** Tab, choose ***PDF Job Options: Folco***, uncheck ***Generate Separate PDF …***, set ***Page Range: All***. On the ***Bookmarks*** Tab activate ***Generate PDF Bookmarks*** and choose the desired paragraphs to include.
- On the ***Tags*** Tab, uncheck ***Generate Tagged PDF*** (to reduce file size).

- On the ***Links*** Tab, uncheck ***Create Named Destinations for All Paragraphs***, to reduce file size.

- Print document to PS file.

- Now you may distill the PS file. Make sure that options in Distiller are as follows. The ***Job Options*** setting must be to Folco, ***Compatibility*** should be Acrobat 3.0 (PDF 1.2).

In this way, the size of both the PostScript file and the final PDF file should be relatively small (like in the previous installation), without too much compromising the quality of results.

Illustrator

The previous version of Illustrator is not compatible with XP. A new product and license will be obtained and installed as soon as possible.

Fixing Adobe program group

The described installation produces separate program start entries in the Start menu: Adobe (containing only FM 7.0), Adobe Acrobat 6.0 and Adobe Distiller 6.0. These may be regrouped under the Adobe folder.

To fix this (as administrator), you must cut the `Acrobat Distiller 6.0` and the `Adobe Acrobat 6.0` folder from `C:\Documents and Settings\AllUsers\Start Menu\Programs` and paste them under `C:\Documents and Settings\All Users\Start Menu\Programs\Adobe`.

You may act similarly also to group all other Adobe applications (e.g., Illustrator) under the same program group.

Fixing file associations

After all these installations, file associations are probably not what one would like to have. This may be fixed (under administrator) and is automatically available to all users. The adjustments are done in Windows Explorer by clicking on ***Tools → Folder Options → File Types***, and by selecting the following file extensions: .PS (for PostScript files, note that this applies automatically also to .EPS, i.e. encapsulated PostScript files), and .PDF.

The suggested settings are:

- .PS file default action: `open`. Application: `"C:\Program Files\Ghostgum\gsview\gsview32.exe" "%1"`. No DDE.

- .PS file action `print`. Application: `"C:\Program Files\Ghostgum\gsview\gsview32.exe" /p "%1"`. No DDE.

- .PS file action `PDF`. Application `"C:\Program Files\Adobe\Acrobat 6.0\Distillr\AcroDist.exe" %1`. No DDE.

- .PDF file default action: `open`. Application: `"C:\Program Files\Adobe\Acrobat 6.0\Acrobat\Acrobat.exe" %1`. No DDE.

- .PDF file action: open with gsview. Application: `"C:\Program Files\Ghostgum\gsview\gsview32.exe" "%1"`. No DDE.

10.12 Version September 2004

On September 2004 a complete re-installation of the mirror site was performed, on a new PC with two processors (3GHz clock rate each) and 1 GB of RAM. The name of the new machine is sm61. All the public-domain tools were re-installed at the same version as was used in March 2004, with the only notable exception of the compilation and development environment.

Two versions of MS Developer Studio and Fortran compiler were installed at the same time:

- Developer Studio 6.0, together with Compaq Fortran compiler 6.6, i.e. the same environment that was active on the previous machine (sm48, March 2004).

- Visual Studio .NET 2003, together with Intel Fortran Compiler 8.0.

The idea is to migrate gradually to the new platform (.NET), while at the same time keeping the possibility to provide support to sites that still use the old platform.

Some malfunctions were detected, as described below.

10.12.1 Slow opening of Developer Studio workspaces

With the dual installation decribed above, .DSW (Workspace) files get associated by default with the .NET platform. Therefore, double-clicking on them opens the new environment which, however, is unable to correctly open a workspace in the previous format!

To open a workspace with the older environment, either click the right button on the .DSW file, the choose "Open with" and "Microsoft ® Developer Studio" (this launches Visual C++ 6.0), or use Start -> All Programs -> Compaq Visual Fortran 6 -> Developer studio (to open Developer Studio proper) and then click on File -> Open Workspace to open the .DSW file. The second method appears to be cleaner.

In either way, opening a previous workspace turned out to be extremely slow. Some investigation revealed that the problem is due to the fact that the old workspace contains at several places the name of the previous machine (sm48). Since this machine is no longer available, probably the system loses a lot of time trying just to access it on the network. Strangely, changing all names sm48 to sm61, saving and re-opening the workspace did not seem to solve the problem.

Finally, the following workaround was found, which may be very useful also in other similar situations. Just edit the two files devel.dsp and eplex.dsp (these are both ASCII files), and replace any occurrences of SM48 by SM61. By saving and opening the workspace again, the problem is solved (quick opening).

10.12.2 DFLIB libraries

Another strange problem that occurred with the dual installation is that, when compiling the IFWIN subroutine for debugging (epx_cmp –w) from the command line an error was encountered (NARGS apparently not declared), while doing the same thing under Developer Studio did not produce any error.

A patch solution was tried: declare INTEGER, EXTERNAL :: NARGS at the beginning of the routine. In this way, command-line compilation works, but Developer Studio compilation fails!

Some further inspection revealed that the problem is related to how the libraries (DFLIB) are sought:

- In the old installation the DFLIB library (dflib.f90, dflib.mod) is located under C:\Program Files\Microsoft Visual Studio\DF98\INCLUDE and contains an adequate declaration for NARGS.

- However, under .NET the DFLIB library (dflib.f90, dflib.mod) is located under C:\Program Files\Intel\Fortran\compiler80\IA32\INCLUDE and does not contains any adequate declaration for NARGS: this routine has in fact become an intrinsic function under the Intel 8.0 compiler.

It should also be noted that after dual installation as described above (I do not know in which order the two environments were installed, however), the environment variable INCLUDE ends up containing the path to the newer platform first, followed by the old one.

Therefore, what happens seems to be the following (with the standard IFWIN.ff, i.e. without any explicit declaration of NARGS):

- When compiling from the command line, the system uses the INCLUDE system variable and ends up using the Intel 8.0 version of dflib.mod: therefore, NARGS is apparently undeclared!

- When compiling under Developer Studio, the system probably does not use the INCLUDE system variable and searches first (or only) the DF98\INCLUDE directory thus finding the "right" version of dflib.mod.

For the moment, I adopt the following temporary workaround solution. Under administrator, set the environment variables as follows:

- In the definition of the INCLUDE variable, move the C:\Program Files\Intel\Fortran\compiler80\IA32\INCLUDE at the end of the string, i.e. after reference to all Microsoft Visual Studio paths.

- In the definition of the LIB variable, move the C:\Program Files\Intel\Fortran\compiler80\IA32\LIB at the end of the string, i.e. after reference to all Microsoft Visual Studio paths.

Note that the second modification listed above (concerning LIB) is also necessary, because if only the first one is performed a strange QuickWin runtime error message appears when launching the QuickWin version of the program (the error is non-blocking, in the sense that by clicking OK the execution continues until the end with no problems, but this behaviour may be annoying).

This workaround seems to fix the development under the old environment, but most probably will be incompatible for use under .NET. When .NET developments will be started, a more rational solution will of course have to be found.

10.12.3 SPLIB package

The SPLIB package, which offers various linear system solvers alternative to the standard Cholesky's method for the treatment of connections, has been added. The package is contained in a library called `Libsp.lib`.

The `epx_cmp` procedure has been modified by adding an extra standard filter keyword (`SPLIB`): at the moment of writing this keyword is used only in file `m_matrix_solver.ff`.

The `epx_lk` procedure has been modified by adding the new library `Splib.lib`. The same applies to Developer Studio projects.

Here is a description of the modifications that have been applied to the SPLIB package (as received from CEA) to port it under the Windows platform (the same text is in file `readme_win.txt` which has been added to the local version of the package).

```
Modifications in the SPLIB package as received from HB on 29/9/2004
to make it work under Windows XP at JRC.
================================================================================

1. The file getr.c is removed from the package.

2. In mytime.f everything, including the line "call getr(...",
   is commented out. We add a line:

     mytime = 0.d0

   so this function always returns 0.

3. The file prtres.f is removed from the package.

4. In splib.f, comment out "call prtres(..." (2 occurrences).

5. The file spdot.c is removed from the package, and replaced by a Fortran
   version of the same function (spdot.f) that reads:
```

```
            REAL(8) FUNCTION spdot (length, x, y, cols)
      *
            IMPLICIT NONE
      *
            INTEGER, INTENT(IN) :: length, cols(*)
            REAL(8), INTENT(IN) :: x(*), y(*)
      *
            INTEGER :: k
      *
            spdot = 0.D0
            DO k = 1, length
              spdot = spdot + y(k)*x(cols(k))
            END DO
      *
            END FUNCTION spdot
```

   In this way there remain no C files in the package and there is no
   need to treat Fortran/C interfacing, which is not portable across
   platforms.

6. Two of the files, namely instr.f and splib.f, contain conditional
   compilation statements "#ifndef cadyf ... #else cadyf ... #endif cadyf".
   In order to avoid precompilation, we assume that cadyf is undefined,
   so we comment out the "else cadyf ... #endif cadyf" branches in
   these two files (one occurrence in instr.f, 3 occurrences in
   splib.f).

```
================================================================================

There remain 5 include (.inc) files and 41 Fortran (.f) files in the package.
Compile everything and produce a library (Libsp.lib) by the following
perl script:

     system ("del *.obj");
     system ("df -c /optimize:5 *.f");
     system ("lib /OUT:Libsp.lib *.obj");
     system ("del *.obj");
     exit;

Move the Libsp.lib library under %EUROPLEXUS%\Library.

Compile EUROPLEXUS module m_matrix_solver.ff with the additional
filter keyword SPLIB (this keyword may be definitely added to the
standard compilation keywords in epx_cmp.pl).

Modify the epx_lk.pl procedure by adding an extra library (Libsp.lib),
then link to produce the executable.

To use the SPLIB model, instead of Cholesky, in the input file type:

    LIAI SOLV CSR ...

instead of just:

    LIAI

In this way, the above modifications may be made permanent in the
EUROPLEXUS environment.
```

10.13 Version August 2005

On August 2005 a complete re-installation of the mirror site had to be performed, on the same machine as previously, because the operating system had to be re-installed due to some unrecoverable malfunctioning. The opportunity was taken to upgrade all development tools as they had to be re-installed anyway.

10.13.1 Operating system

The version of the operating system is still MS Windows XP Professional, Version 2002, but Service Pack 2. The machine is a Pentium 4 with 2 CPUs at 3 GHz and 1 GB of RAM.

10.13.2 Developer Studio and Compilers

Most notably, new versions of MS Developer Studio and Fortran compiler were installed:

- Visual Studio .NET 2003;
- Intel Fortran Compiler 9.0.

The new environment and tools required various modifications to the development tools, scripts and procedures which are detailed below.

10.13.3 Use of the compiler/linker from the command line

During installation of both Visual Studio .NET 2003 and the Intel Fortran Compiler 9.0 the user (administrator) is prompted by dialog windows concerning the "automatic" setting of environment variables (INCLUDE, LIB,. Path and possibly others) to make it possible using the development tools from the command line.

Despite the fact that the corresponding dialog boxes had been checked during installation (at least I strongly believe so …) it turned out that the Fortran compiler (`ifort`) could not be used directly from the command line. On the other hand, the interactive environment seemed to work correctly since the documentation says that it does not use the environment variables.

By using the set command to display the environment, it was confirmed that the appropriate settings for INCLUDE, LIB, Path were missing.

The documentation says that it is possible to set the environment for command line execution in two ways: either by opening the command window from the "***Build environment for IA-32 Applications***" button which appears in Start -> Programs for the Intel compiler (this opens a console window with the environment properly set), or in an already existing console window, by executing the script `ifortvars.bat` (which internally executes also `vsvars32.bat` which sets the .NET variables *before* those for the Fortran compiler).

Both methods do work but they are clearly unusable for unattended, automatic use of the evolution scripts overnight. An attempt was made to execute `ifortvars.bat` from within the `epx_cmp` Perl script, but there are two difficulties: first, the command does not execute correctly after translation of the Perl script to batch file by means of `pl2bat` (unless it is moved to the top of the batch script, before the Perl section, and launched by '`call ifortvars`') and second, upon repeated execution the environment variables continue to be incremented by repeated instances of the directories and so become quickly very long.

Since it was not possible to re-install the Visual Studio and compiler due to long planned absence of the system administrators, it was decided to "patch" the installation by hand. As administrator, the appropriate environment variables were set permanently as System variables, so that they are visible to all users. In this way the above problems are overcome, but the method seems not appropriate e.g. in case of later upgrading of the compilers etc. The setting was done by comparing the output of the set command before and after (interactive) execution of the `ifortvars.bat` script. The result of the `set` command after the patch is listed below, with highlighted the variables which have been added:

```
ALLUSERSPROFILE=C:\Documents and Settings\All Users
APPDATA=C:\Documents and Settings\folco\Application Data
CommonProgramFiles=C:\Program Files\Common Files
COMPUTERNAME=SM61
ComSpec=C:\WINDOWS\system32\cmd.exe
DevEnvDir=C:\Program Files\Microsoft Visual Studio .NET 2003\Common7\IDE
ESOPE_TEMP=C:\TEMP
EUROPLEXUS=E:\EUROPLEXUS
FP_NO_HOST_CHECK=NO
```

```
FrameworkDir=C:\WINDOWS\Microsoft.NET\Framework
FrameworkSDKDir=C:\Program Files\Microsoft Visual Studio .NET 2003\SDK\v1.1
FrameworkVersion=v1.1.4322
GIBI_FILES=C:\Cast3m\gibi_files
GIBI_START=C:\Cast3m\bin
HOMEDRIVE=U:
HOMEPATH=\
HOMESHARE=\\sm61\users\folco
IDB_PATH=C:\Program Files\Intel\
IFORT_COMPILER90=C:\Program Files\Intel\Compiler\Fortran\9.0
INCLUDE=C:\Program Files\Intel\Compiler\Fortran\9.0\Ia32\Include;
        C:\Program Files\Microsoft Visual Studio .NET 2003\VC7\ATLMFC\INCLUDE;
        C:\Program Files\Microsoft Visual Studio .NET 2003\VC7\INCLUDE;
        C:\Program Files\Microsoft Visual Studio .NET 2003\VC7\PlatformSDK\include\prerelease;
        C:\Program Files\Microsoft Visual Studio .NET 2003\VC7\PlatformSDK\include;
        C:\Program Files\Microsoft Visual Studio .NET 2003\SDK\v1.1\include
INTEL_LICENSE_FILE=C:\Program Files\Common Files\Intel\Licenses
INTEL_SHARED=C:\Program Files\Common Files\Intel\Shared Files
LIB=C:\Program Files\Intel\Compiler\Fortran\9.0\Ia32\Lib;
    C:\Program Files\Microsoft Visual Studio .NET 2003\VC7\ATLMFC\LIB;
    C:\Program Files\Microsoft Visual Studio .NET 2003\VC7\LIB;
    C:\Program Files\Microsoft Visual Studio .NET 2003\VC7\PlatformSDK\lib\prerelease;
    C:\Program Files\Microsoft Visual Studio .NET 2003\VC7\PlatformSDK\lib;
    C:\Program Files\Microsoft Visual Studio .NET 2003\SDK\v1.1\lib
LOGONSERVER=\\SIGMA
MIF_PATH=C:\Cast3m\gibi_files
MSVCDir=C:\Program Files\Microsoft Visual Studio .NET 2003\VC7
NUMBER_OF_PROCESSORS=2
OS=Windows_NT
Path=C:\Perl\bin\;
     C:\Program Files\Intel\Compiler\Fortran\9.0\Ia32\Bin;
     C:\Program Files\Common Files\Intel\Shared Files\Ia32\Bin;
     C:\Program Files\Microsoft Visual Studio .NET 2003\Common7\IDE;
     C:\Program Files\Microsoft Visual Studio .NET 2003\VC7\BIN;
     C:\Program Files\Microsoft Visual Studio .NET 2003\Common7\Tools;
     C:\Program Files\Microsoft Visual Studio .NET 2003\Common7\Tools\bin\prerelease;
     C:\Program Files\Microsoft Visual Studio .NET 2003\Common7\Tools\bin;
     C:\Program Files\Microsoft Visual Studio .NET 2003\SDK\v1.1\bin;
     C:\WINDOWS\Microsoft.NET\Framework\v1.1.4322;
     C:\WINDOWS\system32;
     C:\WINDOWS;
     C:\WINDOWS\System32\Wbem;
     C:\Program Files\Intel\IDB\9.0\IA32\Script;
     D:\Users\Folco\Run;
     D:\Users\Folco\Run\Gnu;
     E:\EUROPLEXUS\Util;
     C:\Vim\Vim63
PATHEXT=.COM;.EXE;.BAT;.CMD;.PL;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
PROCESSOR_ARCHITECTURE=x86
PROCESSOR_IDENTIFIER=x86 Family 15 Model 2 Stepping 9, GenuineIntel
PROCESSOR_LEVEL=15
PROCESSOR_REVISION=0209
ProgramFiles=C:\Program Files
PROMPT=$P$G
SESSIONNAME=Console
SystemDrive=C:
SystemRoot=C:\WINDOWS
TEMP=C:\DOCUME~1\folco\LOCALS~1\Temp
TMP=C:\DOCUME~1\folco\LOCALS~1\Temp
USERDOMAIN=SMCM
USERNAME=folco
USERPROFILE=C:\Documents and Settings\folco
VCINSTALLDIR=C:\Program Files\Microsoft Visual Studio .NET 2003
VS71COMNTOOLS=C:\Program Files\Microsoft Visual Studio .NET 2003\Common7\Tools\
VSINSTALLDIR=C:\Program Files\Microsoft Visual Studio .NET 2003\Common7\IDE
windir=C:\WINDOWS
```

10.13.4 WinZip

The evaluation version 9.0 of the WinZip package has been downloaded from the network and installed. This consists of the single file winzip90.exe (click to install). This is the same as in the previous installation.

10.13.5 UnxUtils

The current version of the package has been downloaded and installed in `D:\Users\Folco\Run\Gnu`. The `type.exe` command has been moved to `Gnu_inactive` as usual, to prevent clash with DOS's `type` command.

10.13.6 Netpbm

This package has not been updated and we are still using the previous version. Simply copy the old files to `C:\Netpbm`.

10.13.7 Perl

The new version 5.8.7 (Build 813) consists of the single self-installing file ActivePerl-5.8.7.813-MSWin32-x86-148120.msi, which automatically performs (like the previous one) file association when

installed, thus the manual setting described in Section 3.1.1 is no longer necessary. However, the `PATHEXT` variable setting (Section 3.1.2) is still required.

The behaviour concerning the globbing (expansion of file names) is the same as in the previously installed version (see Section 10.5.1), so no additional modification is required in the procedures.

The Sendmail package (`Sendmail.pm`) has been installed in `C:\Perl\Lib\Mail` and the date package (file `C:\perl\site\lib\HTTP\Date.pm`) has been customized according to the instructions in Section 3.6.

The Perl `head.bat` command (in `C:\Perl\bin`) conflicts with the homonimous Unix-like command which is heavily used in procedures, therefore it is decided, as previously, to disactivate the Perl command by renaming the file `HEAD_ori.bat`.

10.13.8 Vim

The new version 6.3 of the Vim editor (consisting of the self-installing `gvim63.exe` executable) needs the same customization as the previous 6.1 version, described in Sections 10.5.2 and 10.7.7 (and also the 6.2). The configuration file `_vimrc` has been copied from the previous installation, because the base file `C:\Vim\Vim63\vimrc_example.vim` has changed only slightly since the previous version.

The same applies to syntax file `eiffel.vim`. Note that, contrary to what had been stated for the previous customization, the file `fortran.vim` does not need any changes (it was already so in version 6.0). However, `filetype.vim` has changed slightly, so the customization described in Section 10.5.2 has been re-applied to the newer version.

The package installs in `C:\Vim`. The string "`;C:\Vim\Vim63`" has to be added to the `Path` system variable to get universal access to the executables.

10.13.9 Ghostscript and GSview

The current versions are AFPL Ghostscript 8.5.1 (single self-extracting file `gs851w32.exe`) and GSView 4.7 (single self-extracting file `gsv47w32.exe`). Installation consists simply in double-clicking the self-extracting executable files and following the instructions.

By default, Ghostscript installs in `C:\Gs` while GSview (unlike the previous version!) installs in `C:\Program Files\Ghostgum`. Do not forget to checl the "All users" boxes during installation to make the tools accessible to everybody.

10.13.10 PStoEdit

It was decided to not install PstoEdit any more.

10.13.11 Complete code recompilation

Due to changes in the C and Fortran compilers, it was decided to completely recompile the source code and to re-generate the corresponding object libraries (.LIB) and module files (.MOD).

Epx_cmp command

The `epx_cmp` compilation command required only small syntax adjustments due to the new compiler and was updated as follows:

- The keyword `/nooptimize` is replaced by `/optimize:0`;
- The keyword `/noerror_limit` is replaced by `/noerror-limit`;

- The keyword `/compile_only` is replaced by `/compile-only`;
- The name of the compiler, `df`, is replaced by `ifort`.

Furthermore, a new option `-g` had to be added, which causes compilation to occur with the QuickWin libraries (`/libs:qwin` compiler switch). This is now necessary to produce the QuickWin version of the executable (see also the modifications in the `epx_lk` command described below). Compilation with the new option is internally (and automatically) invoked by the `epx_lk` command when the `-w` switch is used, and needs not be invoked directly by the user.

Finally, the script was modified so that, if a compilatiopn error is encountered during a multi-file compilation, the compilation continues with the following files instead of being stopped.

The new script is listed in the Appendix.

Libplex library

With the new `epx_cmp` command the whole code source was recompiled with optimization (`epx_cmp -o`). Initially, a problem occurred when compiling `m_render.ff`, but it was later discovered that this did not depend on the new compiler, but on an error in the epx_ordo procedure that determines the order in which modules have to be compiled. The problem had meanwhile been independently noted at CEA and solved by a corrected version of the `ordo.f` software (see Section 0 below). After installation of the new `ordo.exe`, the compilation went smoothly.

The main library was re-generated by: `LIB /OUT:libplex.lib *.obj` and replaced into `E:\Europlexus\Library`.

The new `.MOD` files were replaced into `E:\EUROPLEXUS\Module`.

BLAS library

The BLAS library was re-generated by the following commands:

```
IFORT /c /optimize:5 libblas.f
LIB /OUT:Libblas.lib libblas.obj
mv Libblas.lib E:\EUROPLEXUS\Library
```

SPLIB library

The SPLIB library was re-generated by the following commands:

```
IFORT /c /optimize:5 *.f
LIB /OUT:Libsp.lib *.obj
mv Libsp.lib E:\EUROPLEXUS\Library
```

Libplex_c library

The Libplex_c library was re-generated by the following commands:

```
cl -c alloc.c
cl -c ctri.c
LIB /OUT:Libplex_c.lib *.obj
mv Liplex_c.lib E:\EUROPLEXUS\Library
```

With the above modifications, the entire source of EUROPLEXUS has been re-generated, with the notable exception of all parts related to the OpenGL graphical module, namely the following libraries:

- the dynamically linked run-time library `glut32.dll` (for the modified GLUT package).
- `f90gl.lib`, `f90glu.lib`, `f90glut.lib` (containing the 'F90GL' Fortran bindings to OpenGL, GLU and GLUT, respectively);
- `bmplib.lib` (for bitmap image and animations creation);

An attempt was first made to link the code without updating the above libraries. However, this resulted in a large number (634) of unresolved externals (in practice the whole set of OpenGL-related functions) apparently due to the fact that the new compiler uses a different naming of external functions (an underscore followed by the function name in uppercase).

After some unsuccessful tests with compiler switches, it was decided to re-generate also the OpenGL-related libraries.

GLUT library

It was checked that the GLUT library is still currently in the version 3.7.6 (8 November 2001) as previously, consisting in the file `glut-3.7.6-src.zip`. To the source, various modifications as described in reference 12 have to be applied, which are contained in the package `Glut_f90gl_modifs.tar.gz` by the author.

The 3.7.6 package contains a Visual Studio project (`GLUT.DSW`) that may be opened with the new version of Visual Studio and is automatically converted to the new formal (`GLUT.SLN`). However, the automatic translation process leaves the "Post-build step" action with the old paths for the lib and includes, which are no longer valid. These have to be modified as follows:

```
copy $(TARGETDIR)\glut32.dll %WINDIR%\SYSTEM32
copy $(TARGETDIR)\glut32.lib "$(MSDevDir)\..\..\VC98\lib"
copy ..\..\include\GL\glut.h "$(MSDevDir)\..\..\VC98\include\GL"
```

becomes:

```
copy $(TARGETDIR)\glut32.dll %WINDIR%\SYSTEM32
copy $(TARGETDIR)\glut32.lib "$(DevEnvDir)..\..\VC7\PlatformSDK\lib"
copy ..\..\include\GL\glut.h "$(DevEnvDir)\..\..\VC7\PlatformSDK\include\GL"
```

where the `DevEnvDir` system variable has the value:

```
DevEnvDir=C:\Program Files\Microsoft Visual Studio .NET 2003\Common7\IDE
```

After re-building the project, do not forget to copy *by hand* the file

```
glut-3.7.6\include\GL\glutf90.h
```

into the directory

```
C:\Program Files\Microsoft Visual Studio .NET 2003\VC7\PlatformSDK\include\GL
```

(where also `glut.h` resides), since this action is not contained in the Post-Build step, as already remarked in reference 12. The include is needed in the building of F90GL.

F90GL libraries

An attempt of recompiling the old version of the F90GL package (modified as explained in reference 12) with the new compiler gave no errors, but when using the resulting libraries in linking EUROPLEXUS, the above-mentioned problem of 634 unresolved externals, due to the leading underscore problem in external function names, was encountered. Various attempts to correct the source failed.

It was found, however, that a new version of the F90GL package (1.2.8 instead of 1.2.4) has meanwhile become available, and this version contains a specific compilation script for the new compiler (actually for Intel 8.0 fortran, but it's OK), called `mf8nio.bat` instead of `mf8nvo.bat` (this was the version for the Digital/Compaq 6.5 Fortran compiler). It was therefore decided to switch to the new version.

The new sources were obtained as file `f90gl128.zip`. To these sources, the following modifications had to be (re-) applied:

- The "parent" compilation script, now `mf8nio.bat`, (in directory `f90gl-1.2.8`) was modified by changing the value of the `WININC` variable from: `set WININC=F:\Program Files\Microsoft SDK\include` to: `set WININC=c:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\PlatformSDK\Include`;

- The file `f90gl-1.2.8\glut\cwrapglt.c` was modified by re-applying the modifications described in reference 12. At the same time, a redundant printout "`fortranWMCloseWrapper entered …`" was removed, which appeared on the EUROPLEXUS console each time the graphical window was closed.

- The file `f90gl-1.2.8\glut\fwrapglt.fpp` was modified by re-applying the modifications described in reference 12;

- The file `f90gl-1.2.8\glut\intrfglt.fpp` was modified by re-applying the modifications described in reference 12;

Then, the package was re-compiled and installed as follows:

- Execute the `mf8nio.bat` script, which produces three `.LIB` files and eight `.MOD` files in the directory `f90gl-1.2.8\lib`;

- Copy by hand the three `.LIB` files on the directory:
  `C:\Program Files\Intel\Compiler\Fortran\9.0\IA32\Lib`;

- Copy by hand the eight `.MOD` files on the directory:
  `C:\Program Files\Intel\Compiler\Fortran\9.0\IA32\Include`

Bmplib library

An attempt of recompiling the library as-is gave a C compilation error in the file `bitmap.c`. The new C compiler complains about different definitions of the `exit()` function (by the way, this function is *not* explicitly used in `bitmap.c`) in the two includes: `C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\Include\stdlib.h` (line 256) and `C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\PlatformSDK\Include\gl\glut.h` (line 146).

A workaround solution was found by moving the inclusion of the `bitmap.h` header *after* `stdlib.h` in file `bitmap.c` (like it is in the companion source file `bmpsave.c`). In other words, at the beginning of `bitmap.c` the lines:

```
#include "bitmap.h"
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
```

become:

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include "bitmap.h"
```

With these modifications, the library was built correctly.

Epx_lk command

The `epx_lk` linkage command was adapted to conform to the new compiler and linker. There were two substantial changes necessary:

- First, the `link` command is now used directly, instead of the compilation command (`df` or `ifort`) with the `/LINK` option. This is to avoid problems with the way the (new) compiler transmits options to the linker (some keywords previously used are no longer available).

- Second, the production of the QuickWin version of the executable has been considerably changed. It turned out in fact that now ***all source files in the current directory*** (and not only those using QuickWin function directly, i.e. `m_qwin` and `ifwin`) have to be (re-)compiled with the `-g` compiler option (`/libs:qwin`), else there are link errors due to unresolved references.

Thus the linking strategy in case a QuickWin version is requested (`epx_lk -w ...`) has been changed as follows:

- Retrieve files `m_qwin.ff` and/or `ifwin.ff` and/or `main.ff` in case they are not present on the current directory (for the console case, only `main.obj` is retrieved, and from the object library);

- (Re-)compile all sources in the present directory with the `-g` option to make them 'compatible' with QuickWin. This is now done by calling the `epx_cmp` command without any file names: as a consequence, files are compiled in the correct order whether or not an epx_ordo file is present (see `epx_cmp`).

- Re-compile `m_qwin.ff` and `ifwin.ff` with the `-g` option ***and*** the additional keyword `QWIN` for the filtering process.

- Execute the link command with appropriate switches to produce the QuickWin executable (this command is similar to the one implemented previously, but `df` has been replaced by `link`).

- Remove any files that were retrieved explicitely to produce the QuickWin version. An important change is that all object files are now removed (in case of QuickWin version), because, due to the re-compilation with `-g` option mentioned above they are probably no longer usable for other (successive) links.

The latter point above implies that the `epx_lk -w` command (QuickWin version) should be used with care. Be aware that your local object files are destroyed after a link with this option. This is not a problem in the `epx_evol_start` procedure since the QuickWin executable is built at the end of the procedure, i.e. ***after*** the console executable.

The new script is listed in the Appendix.

Includes evolution

As a consequence of the new procedure needed to produce the QuickWin version and outlined above, the strategy adopted for an evolution in the presence of a package of modified includes had to be updated.

The problem was that the `epx_evol_includes` script (called internally by `epx_evol_start`) retrieves the sources of "includer" files (i.e. all the sources using the modified includes), compiles them normally and then deletes the retrieved sources by leaving the object and module files on the evolution directory. Then, when the QuickWin version is produced, these object files cause errors in the link because they have been produced without the `-g` compiler option.

Thus, it was necessary to substantially review some of the evolution procedures, namely `epx_test_includes`, `epx_test_sources` and `epx_evol_start` itself. The main modifications and ameliorations are shortly described below.

Epx_evol_start script

The new strategy for an evolution in the presence of an includes and/or a sources package may be outlined as follows:

- If an includes package is present in the evolution, it is unpacked. Then the "includers", i.e. all source files that use the modified includes, are retrieved from the sources directory. This is done by a new procedure `epx_get_includers` (see description below) which replaces the former `epx_test_includes`.

- If a sources package is present in the evolution, it is unpacked (note that this operation may overwrite some of the previously retrieved includers, if any). Then, if an includes package is also present, the `epx_ordo.txt` file usually present in the sources distribution package is

deleted: this is necessary because this file will contain only the list of the modified sources, and not that of the includers. In any case, a fresh and complete `epx_ordo.txt` file will be produced upon the following sources compilation (see below).

- Any sources present are compiled, by invoking `epx_cmp` directly, without any file arguments. In this way an `epx_ordo.txt` file is created anew if not yet present. The former `epx_test_sources` script becomes obsolete (its functionality is already conatined in `epx_cmp` itself).

- The standard executable is produced, and benchmarks (first any modified ones, then the standard ones) are executed.

- The following libraries are updated in sequence, as appropriate: includes, sources, modules, objects. Then the standard executable is updated. Note that in the case of includes evolution the retrieved includers will replace the former ones, unlike in the previous strategy. For sources, this is not a problem since the files are unchanged

- The QuickWin executable is produced by invoking `epx_lk -w`. Since any retrieved includers sources are still present, they are compiled as well, thus avoiding link errors. Furthermore, sources are now compiled just once, and in the correct order, thanks to invocation of the `epx_cmp` command without any file arguments from within the `epx_lk -w` procedure. This executable is updated.

- The benchmarks are updated if appropriate.

- The manuals are processed and updated if needed.

- The validations are processed and updated if needed.

- Any history files are processed and updated. Note that in the case of inclkudes evolution the only history files modified are those of the effectively changed sources (as contained in the sources distribution package). The history files of the retrieved includers are not modified.

- Clean up and exit.

Epx_test_includes script

This procedure is now obsolete and is replaced by `epx_get_includers`, described below.

Epx_get_includers script

This procedure replaces the former `epx_test_includes`. It builds a list of all include files present in the current directory. Then, it retrieves from the sources directory all the source files that use any of the present includes.

There is no compilation and removal of files, unlike in the previous implementation. Therefore, this command may now be useful also during the normal development of the code, and not only via `epx_evol_start`.

Epx_test_sources script

This procedure is now obsolete. Its functionality is now embedded in `epx_cmp`, described above

10.13.12 Correction of the epx_ordo command

A problem was noticed in the functioning of the `epx_ordo` command due to the recent proliferation of module packages, which might lead in some cases to failure in the compilation of `m_render.ff` when recompiling the whole code from scratch.

CEA corrected the problem and provided a new source (`ordo.f`) for the executable `ordo.exe`, which has been implemented succesfully with minor cosmetic changes and is listed in the Appendix.

10.13.13 Correction of the epx_evol_histories procedure

A problem was noticed in the functioning of the `epx_evol_histories` procedure, which sometimes failed in updating the local history files.

The cause was identified in the `fold` command (part of the Gnu utilities) which has been updated in the present installation and sometimes adds an extra carriage return <CR> character at the end of each line of the folded file, thus preventing the subsequent pattern matching from working correctly. Note that this character immediately preceeds the final newline character <NL> which is normally placed at the end of each line.

The correction consisted in simply filtering each line after folding, by removing the final carriage return <CR> if present.

10.13.14 MiKTeX

The current MiKTeX package is still Version 2.4 like in the previous implementation. However, since minor changes did occurr in the meantime (now the package minor version is 1705) it was decided to download the current version from the Web rather than re-installing the previous one. The installation file is now a single self-installing file `small-miktex-2.4.1705.exe` corresponding to a "small" MiKTeX system (about 25 MB).

After installation, it is no longer necessary to perform the file association of `.dvi` files with the `yap` previewer, unlike with the previous version 2.2.

The same customizations described in Sections 7.5.9 and 7.5.10 have been re-applied. The old versions of the non-standard LaTeX packages (see Table in Section 7.5.9) have been re-used (except `hevea.sty`, see below). However, the customization of file `miktex.ini` has been re-applied on the newer version of this file, which had slightly changed since the previous installation.

Since the `fancyhdr` packege seems to be no longer part of the standard MiKTeX distribution (unlike in the previous standard distribution), the style files from a previous implementation were used: this consists simply in copying `fancyhdr.sty` and `extramarks.sty` (this latter one is perhaps redundant) to `C:\Texmf\Tex\Latex\Folco`.

10.13.15 Hevea

An attempt was made to use the new version of the `hevea` package (1.08). Unfortunately, already in the standard testing phase of the package (as described in Section 10.5.4) the result of command `hevea pavtest.tex` was quite different from what expected (title and parentheses were missing). Therefore, the previous version 1.06 was reused (since version 1.07 also had a bug concerning the index, as already noticed in Section 10.11.12). Installation was performed as described in Section 10.5.4 for that version.

10.13.16 Tth

The `tth` package is no longer contained in the standard MiKTeX distribution. Use the old package. Installation is done as explained in Section 10.5.5.

10.13.17 Installation of Adobe products

Installation of the various Adobe products was made like in the previous implementation (see Section 10.11.14), with the following slight differences:

- The Acrobat 6.0.4 Professional and Acrobat Distiller 6.0.1 packages were installed first, as part of the machine standard setup, by the system administrator.

- All following installations were done as administrator, as previously suggested.

- Next, the `hp4000_ps` printer driver was installed.

- Then, the previous version of the Adobe PostScript Driver (Acrobat Distiller) was installed exactly as described in Section 10.11.14 under "Adobe PostScript Driver". This "printer" was temporarily set as the default printer for the following installation of FrameMaker.

- Then, FrameMaker 7.0 was installed as described in Section 10.11.14 under "FrameMaker 7.0".

- Finally, settings for PDF output were adjusted as described in Section 10.11.14 under "Settings for PDF output", and the Adobe program group and file associations were also adjusted.

Testing revealed some malfunctionings which, however, may be solved as follows:

- When one oprns a FrameMaker document and used PDF Setup, the setting "PDF Job Options: Folco" appears as default, but when the box is closed an error message appears saying that "the chosen job option does not exist". Despite this message, it seems that any changed settings are retained by the system, so simply ignore this message.

- To visualize or print FrameMaker documents on various "printers" (including files), sometimes font problems are experienced. A particularly frequent one is the mathematical '-' (minus) sign which is transformed into a sort of '<' symbol! To avoid this behaviour a simple strategy may be as follows: **never change printer from within FrameMaker**. If you want to change the printer destination for a file, then: 1) save it and close FrameMaker, then 2) **change your default printer to the desired one**, and finally 3) re-open the file with FrameMaker and print it. Usually in this way the file both visualizes (on screen) and prints correctly (despite the printer that was active when the file was created).

10.13.18 Animations quality problem in PowerPoint

A problem with animations (.AVI files) embedded in PowerPoint documents has been experienced. The version of PowerPoint has changed in this installation and is now PowerPoint 2003 SP1. The problem consists in bad animation (upper part of the frame is somewhat obscured) playing and occurs both with old .PPT files (produced with previous versions of PowerPoint) and with freshly created ones. It is under investigation.

10.13.19 Preparing an EUROPLEXUS solution under .NET 2003

The new Intel 9.0 compiler comes with a command-line debugger `idb` which, unlike in the previous versions, allows to debug the code without resorting to the window-based debugger. An advantage of this debugger is that it should work in pretty the same way under the various supported platforms (e.g. under Linux besides under MS-Windows).

However, the interest of being able to run and debug the code in a visual way under the window-based **MicroSoft Visual Studio .NET 2003** environment is clear and so it was decided to update the procedures described in Section 8 to the new version of the compiling environment.

Actually, the .NET 2003 is able to open and automatically convert the previous version of Developer Studio Workspace. However, it was found that the automatically generated "solution" (this is the new name of the workspace) is not optimal, and thus it was decided to re-build solutions (both the console-based OpenGL version and the QuickWin version) from scratch under .NET 2003.

In order to develop EUROPLEXUS under Microsoft Visual Studio .NET 2003, set up a new Solution (say `plex`) containing two Projects: `filter` and `epx` (formerly called `devel` and `eplex`,

respectively). The `filter` project contains the unfiltered "`*.ff`" Fortran source files, while the `epx` project contains the corresponding filtered sources "`*.f`", which may actually be compiled. The filtering process is automatically performed since it is set up as a "custom build step" action. Hereafter, we describe the set up of a Solution of type Console application, which is the standard for EUROPLEXUS (since May 2003). The set up of a QuickWin application would be slightly different, and the variations are marked like this in parentheses when appropriate.

To set up the Solution, do the following:

1. **CREATE AN EMPTY SOLUTION** - Open Microsoft Visual Studio .NET 2003 (***Start → All Programs → Microsoft Visual Studio .NET 2003 → Microsoft Visual Studio .NET 2003***). This opens up the "Microsoft Development Environment [design] – Start Page" window. Click on menu ***File → New → Blank Solution***. This opens up the New Project dialog box with the ***Visual Studio Solutions → Blank Solution*** already selected.

   The same result may be obtained by activating the ***Start Page → Projects*** pane (not ***Start Page → Online Resources*** nor ***Start Page → My profile***), by clicking on ***New Project*** (which opens up the New Project dialog box) and then by manually choosing ***Visual Studio Solutions → Blank Solution***.

   In any case the "New Project" dialog box opens up, ready to "Create an empty solution containing no projects". If you see a button marked ***More***, click on it so it changes to ***Less***: this will visualize the New Solution Name edit box.

   Under ***Name*** type in `plex`, and under ***Location*** type your directory of choice (e.g.: `E:\Wutemp\Intel90`) and then click on ***OK***. This creates a (new) sub-directory `E:\Wutemp\Intel90\plex` containing the solution file `plex.sln` and an associated file `plex.suo` (the latter appearing as grayed out).

   Click on ***File → Save All*** to save the work done so far (do this from time to time).

2. **ADD A "FILTER" PROJECT TO THE SOLUTION** - A "Solution Explorer" pane appears in the right upper part of the application window, that contains "Solution 'plex' (0 projects)". Right-click on this sentence to show a pop-up menu and click on ***Add → new project***.

   The "Add New Project" dialog appears, under Project Types choose ***Visual C++ Projects → .NET → Empty Project (.NET)***, in the ***Name*** box type `filter` and in the ***Location*** box leave `E:\Wutemp\Intel90\plex` which should appear already as such.

   By clicking on ***OK***, the new project `filter` should appear in the Solution Explorer under "Solution 'plex' (1 project)", with four sub-entries (empty for the moment): References, Source Files, Header Files and Resource Files. A new file `plex.ncb` and a new sub-directory `filter` are created under `E:\Wutemp\Intel90\plex`. The latter should contain a file called `filter.vcproj`.

   Right-click on the `filter` project in the Solution Explorer to show the pop-up menu and choose ***Properties***. This opens up the "filter Property Pages" dialog. In the upper part of the dialog, under ***Configuration***, make sure that you choose ***All Configurations***, so that the subsequent settings will apply to both standard configurations of the project (Debug and Release): although only the Debug configuration will actually be used for this project, it is better to avoid confusion.

   Under ***Configuration Properties → Project Defaults → Configuration Type*** appears "Application (exe)". This entry should be changet into "Utility": to do so, click on it until a drop-down menu appears (a down arrow) and then choose ***Utility*** from the menu. Then click on the ***Apply*** button. In this way all the entries (Linker, Web deployment, etc.) that were previously present under Configuration Properties should disappear and the only remaining ones should be: General, Debugging and Build Events. Click on ***OK*** to close the dialog.

   ADD FILES TO THE FILTER PROJECT - Now let's us add an (unfiltered) source file to the `filter` project. Open a Console window on directory `E:\Wutemp\Intel90\plex` and type

command `epx_get main`: this will copy the EUROPLEXUS main program `main.ff` from the sources library to this directory. Next, in the Solution Explorer, right-click on ***filter → Source Files*** to make the pop-up menu appear, and choose ***Add → Add Existing Item***. The "Add existing Item – filter" dialog appears, already set with ***Look in: filter***. Change it to the parent directory by clicking on the appropriate icon (a folder icon with an upward arrow) so that it becomes ***plex***, and under ***Files of type*** choose ***All Files (*.*)***. Alternatively, under ***File name*** you may enter ****.ff***, so that only the EUROPLEXUS source files are visualized. Select `main.ff` by clicking on it, then click the ***Open*** button. The file `main.ff` should now appear under ***filter → Source Files*** in the Solution Explorer. (QuickWin version: add also files `ifwin.ff` and `m_qwin.ff` to the `filter` project)

SET THE FILTERING PROPERTIES - Next, we must set the filtering for the newly added file. Right-click on the file name (`main.ff`) in the Solution Explorer (QuickWin version: select all three files `main.ff`, `ifwin.ff` and `m_qwin.ff`) to open the pop-up menu, and choose ***Properties***. This opens the "main.ff Property Pages" dialog. Under ***Configuration***, make sure you choose ***All Configurations***, then click under ***Configuration Properties → Custom Build Step*** and under ***Command Line*** insert the following string:

```
%UTILDIR%\epx_filbat.bat "$(InputDir)""$(InputName)"
```

Note that there is a space between 'bat' and the first double quote but there should be no space between the two contiguous double quotes in the above command! For the meaning of the `UTILDIR` environment variable, see Section 10.13.23 below.

In the QuickWin case, instead:

```
%UTILDIR%\epx_filbat_quickwin.bat "$(InputDir)""$(InputName)"
```

The `epx_filbat.bat` file contains the following simple commands:

```
@echo off
echo Filtering %1.ff
%UTILDIR%\epx_filter.exe WIN OGL %2 %3 <%1.ff >%1.f
echo Filtering done
```

The `epx_filbat_quickwin.bat` file is similar, except that it uses the additional compilation keyword `QWIN`.

Under ***Outputs***, insert:

```
$(InputDir)$(InputName).f
```

Then click on the ***Apply*** button and finally on the ***OK*** button to close the dialog.

This has the effect of invoking the `epx_filter` utility described in Section 5.1 with the two (hard-coded) passwords `WIN OGL` on file `$(InputDir)\$(InputName).ff` and placing the filtered output in file `$(InputDir)\$(InputName).f`. Here `$(InputDir)` is `E:\Wutemp\Intel90\plex` and `$(InputName)` is for example `main`.

Having selected multiple Source Files entries (by shift-clicking them) rather than a single file (`main.ff`) when applying the Custom Build setting, has the effect that the custom build action is associated to all files currently selected. This is important to speed up the operation when many files are present in the project. However, unfortunately the operation has to be repeated anew whenever new files are added to the project.

TEST OUT THE FILTERING PROCESS - To test out the correct functioning of the `filter` project, right-click on the `main.ff` file under Solution Expolorer, and from the pop-up menu choose ***Compile***. This should trigger the filtering of the file to produce `main.f`, that will be located in the same directory as `main.ff`, i.e. under `E:\Wutemp\Intel90\plex`. A log of the filtering process should appear in the ***Output*** pane of the application window and also as a file

named `Buildlog.htm` in the `filter\Debug` sub-directory. The messages appearing in the Output pane should be similar to the following:

```
------ Build started: Project: filter, Configuration: Debug Win32 ------

Performing Custom Build Step
Filtering "e:\WUTemp\Intel90\plex\""main".ff
1
Filtering done

Build log was saved at "file://e:\WUTemp\Intel90\plex\filter\Debug\BuildLog.htm"
filter - 0 error(s), 0 warning(s)


---------------------- Done ----------------------

    Build: 1 succeeded, 0 failed, 0 skipped
```

3. **ADD A COMPILATION PROJECT (EPX) TO THE SOLUTION** - It is now time to add the second project to the solution, i.e. the `epx` project, that will perform the actual compilation and link to produce the EUROPLEXUS executable.

   In the Solution Explorer, right-click on the "Solution 'plex' (1 project)" to show the pop-up menu and click on ***Add → new project***. The "Add New Project" dialog appears, under Project Types choose ***Intel® Fortran Projects → Console Application*** (QuickWin version: choose ***Intel® Fortran Projects → QuickWin Application***), in the ***Name*** box type `epx` and in the ***Location*** box leave `E:\Wutemp\Intel90\plex` which should appear already as such.

   By clicking on ***OK***, the "Fortran Console Application Wizard – epx" dialog appears ready to create a project with main program sample code, which is not what we want. Click on the ***Application Settings*** button and select ***Console application type → Empty project*** instead, then click on the ***Finish*** button.

   (QuickWin version: by clicking on ***OK***, the "Fortran QuickWin Application Wizard – epx" dialog appears ready to create an empty QuickWin project with multiple windows, which is exactly what we want. Click on the ***Finish*** button.)

   A new project `epx` should appear in the Solution Explorer pane, which should change to "Solution 'plex' (2 projects)". The new project contains three (empty) sub-entries: Source Files, Header Files and Resource Files. A new sub-directory `epx` is created under `E:\Wutemp\Intel90\plex`. This should contain a file called `epx.vcproj`.

   ADD FILES TO THE EPX PROJECT - Now let's us add the previously obtained (filtered) source file `main.f` to the `epx` project. In the Solution Explorer, right-click on ***epx → Source Files*** to make the pop-up menu appear, and choose ***Add → Add Existing Item***. The "Add existing Item – filter" dialog appears, already set with ***Look in: epx***. Change it to the parent directory by clicking on the appropriate icon (a folder icon with an upward arrow) so that it becomes ***plex***, and make sure that under ***Files of type*** the ***Common Intel® Fortran Files*** type is selected (it should be automatically so). Alternatively, under ***File name*** you may enter ****.f***, so that only the EUROPLEXUS filtered source files are visualized. Select `main.f` by clicking on it, then click the ***Open*** button. The file `main.f` should now appear under ***epx → Source Files*** in the Solution Explorer. (QuickWin version: add also files `ifwin.f` and `m_qwin.f` to the `epx` project).

   SET PROJECT DEPENDENCIES - Finally, we should specify that the `epx` project "depends" upon the `filter` project. Select the menu ***Project → Project Dependencies*** and under the Dependencies tab in the Projects selector select `epx`, then in the Depends on selector check out the box corresponding to the `filter` project. Conversely, the `filter` project should not depend upon `epx` (otherwise a circular dependency would be created), so the corresponding check box should appear grayed out. In the Build Order pane, make sure that "Projects build in this order" contains first `filter` and then `epx`. Then click on the ***OK*** button to close the dialog.

4. **ADJUST THE SETTINGS FOR THE EPX PROJECT** - Right-click on the `epx` project in the Solution Explorer to show the pop-up menu and choose ***Properties***. This opens up the "epx Property Pages" dialog. In the upper part of the dialog, under ***Configuration***, make sure that you choose ***All Configurations***, so that the subsequent settings will apply to both standard configurations of the project (Debug and Release): although the Debug configuration will most often be used for this project, it is better to avoid confusion.

   Under ***Configuration Properties → Fortran → General*** set Additional Include Directories to the following:

   ```
   $(EUROPLEXUS)\Include;$(EUROPLEXUS)\Module
   ```

   and then click on ***Apply***. This has as an effect that include files (`*.inc`) and object module files (`*.mod`) which are not found on the current directory are searched in the two listed directories before the default directories.

   Under ***Configuration Properties → Fortran → Data*** set "Local Variable Storage" to ***Local Variables Automatic*** instead of ***Default Local Storage***. Click on ***Apply***.

   Under ***Configuration Properties → Fortran → Run-time*** set "Generate Traceback Information" to ***Yes*** instead of the default (which appears to be Yes for the Debug configuration and No for the Release configuration). Click on ***Apply***. This generates an interactive warning message which may be ignored.

   Under ***Configuration Properties → Fortran → Run-time*** set "Check Array and String Bounds" to ***No*** instead of the default (which appears to be Yes for the Debug configuration and No for the Release configuration). This is necessary due to the variable passing strategy of EUROPLEXUS coming from its ancestors). Click on ***Apply***.

   Under ***Configuration Properties → Linker → General*** set "Additional Library Directories" to the following:

   ```
   $(EUROPLEXUS)\Library
   ```

   and then click on ***Apply.***

   Under ***Configuration Properties → Linker → Input*** set "Additional Dependencies" to the following:

   ```
   Libplex.lib Libplex_c.lib Libblas.lib Libsp.lib f90gl.lib
   f90glu.lib f90glut.lib bmplib.lib user32.lib gdi32.lib vfw32.lib
   ```

   and then click on ***Apply.***

   Under ***Configuration Properties → Linker → System*** set "Stack Reserve Size" to ***6553600*** instead of the default value of 0 and then click on ***Apply***.

   All settings so far were applied to all configurations, i.e. both Debug and Release configurations. Next, we apply some configuration-specific settings.

   If one builds the Release configuration with the settings made so far, it succeeds without any error messages. However, when building the Debug configuration, the following link warning is issued:

   ```
   LINK : warning LNK4098: defaultlib 'LIBC' conflicts with use of
          other libs; use /NODEFAULTLIB:library
   ```

   The executable (`epx.exe`) is produced despite the warning, but it may be desirable to avoid the message. To this end, the following setting must be applied to the Debug configuration only.

   Right-click on the `epx` project in the Solution Explorer to show the pop-up menu and choose ***Properties***. This opens up the "epx Property Pages" dialog. In the upper part of the dialog, under ***Configuration***, make sure that you choose ***Active(Debug)***, so that the subsequent settings will apply only to the Debug configuration.

Under ***Configuration Properties → Linker → Input*** set "Ignore Specific Library" to ***libc.lib***, click on ***Apply*** and then on ***OK***.

(QuickWin version: the ***libc.lib*** library must be ignored in both configurations, not only in the Debug configuration).

5. **DEFINE THE STARTUP PROJECT** - Next, we must set the `epx` project as the startup project for the solutioin, so that it is executed when debugging is launched. Right-click on "Solution 'plex' (2 projects" and select ***Set Startup Projects***. The "Solution 'plex' Property Pages" dialog opens up.

   In the ***Common Properties → Startup Project*** pane set the Single Startup Project radio button and select `epx` (not `filter`) as the startup project, click on ***Apply*** and then on ***OK***.

6. **TEST OUT THE SOLUTION** - The setting of the Solution is now complete. To test it, first remove the filtered file `main.f` from the `E:\Wutemp\Intel90\plex` directory (to force its re-generation), then click on "Solution plex (2 projects)" to highlight the whole solution, make sure the current configuration is Debug, and finally choose ***Build → Rebuild Solution***. This should rebuild the entire solution, i.e. the two projects `filter` (filtering of `main.ff`) and `epx` (compilation of `main.f` and link). The messages should be something like:

```
------ Rebuild All started: Project: filter, Configuration: Debug Win32 ------
Deleting intermediate files and output files for project 'filter', configuration
        'Debug|Win32'.
Performing Custom Build Step
Filtering "e:\WUTemp\Intel90\plex\""main".ff
1
Filtering done
Build log was saved at "file://e:\WUTemp\Intel90\plex\filter\Debug\BuildLog.htm"
filter - 0 error(s), 0 warning(s)
------ Rebuild All started: Project: epx, Configuration: Debug Win32 ------
Deleting intermediate files and output files for project 'epx', configuration
        'Debug|Win32'.
Compiling with Intel Fortran 9.0...
..\main.f
Linking...
Build log written to  file://E:\WUTemp\Intel90\plex\epx\Debug\BuildLog.txt
epx build succeeded.
---------------------- Done ----------------------
    Rebuild All: 2 succeeded, 0 failed, 0 skipped
```

   A new executable `epx.exe` should be produced in directory `epx\Debug`. This directory will also contain the object file `main.obj`, and a couple of auxiliary files (`epx.pdb`, `Buildlog.txt`). Another log file `Buildlog.htm` appears in directory `filter\Debug`.

   Repeat the above build process also for the Release configuration. This should also work smoothly.

The setup of the Solution is now complete. Make sure you save your work by clicking on ***File → Save All*** before quitting the application.

10.13.20 Using the solution under .NET 2003

Once prepared an EUROPLEXUS development solution as detailed in the previous Section (or, more efficiently, by using the command presented in Section 8.3 or 10.13.21), its use generally consists in adding new or modified source files to the `filter` and `epx` projects, in re-generating the solution by performing the necessary filterings, compilations and link, and in running the test code through the debugger.

It is essential to note that any modifications to source code should be exclusively performed in the unfiltered sources, i.e. to files `.ff`. The filtered versions of these files (`.f`) are to be considered as intermediate files that should never be edited by hand.

Adding a source file to the solution consists in the following steps:

- Edit the file from scratch if it is new, else retrieve the current version of the file from the EUROPLEXUS source library (by the `epx_get` command) and edit it to perform the necessary modifications. In any case, a new file, say `celem.ff`, appears in the development directory.

  Add the `.ff` file to the `filter` project: in the Solution Explorer, right-click on ***filter → Source Files*** to make the pop-up menu appear, and choose ***Add → Add Existing Item***. The "Add existing Item – filter" dialog appears, already set with ***Look in: filter***. Change it to the parent directory by clicking on the appropriate icon (a folder icon with an upward arrow) so that it becomes ***plex***, and under ***Files of type*** choose ***All Files (*.*)***. Alternatively, under ***File name*** you may enter ****.ff***, so that only the EUROPLEXUS source files are visualized. Select `celem.ff` by clicking on it, then click the ***Open*** button. The file `celem.ff` should now appear under ***filter → Source Files*** in the Solution Explorer. In case of multiple files to be added, multiple selection may be done (by shift-click) so that they are all added to the project in one step.

- Set the Custom Build action for the `celem.ff` file (if multiple files have been added, the same operation may be applied to all of them at a time by means of multiple selection). Right-click the `celem.ff` icon under Source Files in the `filter` project, choose ***Properties*** from the pop-up menu and select the ***Custom Build*** tag. The ***Command Line*** and ***Outputs*** fields are empty. The values from any previously set file must be copied: in the present example, the only available file is `main.ff`.

  Right-click the `main.ff` icon under Source Files in the `filter` project, choose ***Properties*** from the pop-up menu and select the ***Custom Build*** tag. Highlight and copy (`CTRL+C`) the contents of the ***Command Line*** field. Then click on the `celem.ff` icon, paste the contents into the ***Command Line*** field (`CTRL+V`) and click on ***OK***. Repeat the same sequence for the ***Outputs*** field.

  When performing the above settings, make sure that in the ***Project Settings*** dialog, the ***Settings For*** field contains either All Configurations or Win32 Debug (if this is the only configuration). In this way, the (same) filtering command will be active for all configurations.

  This procedure is similar to the one for the previous version of Developer Studio, except for the fact that in order to select all files in the project one may no longer select the Source Files entry but must hichlight all the concerned file names by multiple selection: click on the first one and then shift-click on the last one to select them all.

- Filter the `celem.ff` file to produce `celem.f`. This may be accomplished e.g. by right-clicking on the `celem.ff` icon under the Source Files entry of devel files, and then by clicking on ***Compile***.

- Add the filtered file `celem.f` to the `epx` project. Click on the Source Files entry of `epx` files, press the right mouse button and click on ***Add → Add Existing Item***. Select the `celem.f` file and click ***OK***.

- Re-build the `epx` project. Right-click on the `epx` project in Solution Explorer, then in the pop-up menu click on ***Build*** or ***Rebuild***. The compilation of `celem.ff` followed by a link should occur (no filtering is done because the filtered version of `celem` should be up to date).

10.13.21 Generating the solution automatically

The perl script `epx_init` presented in Section 8.3 has been updated. This command automatically creates a new EUROPLEXUS development workspace in the current directory. This command is obviously available to all users and developers of EUROPLEXUS. The syntax is:

```
epx_init [-w]
```

where:

`-w` Build a QuickWin application workspace instead of a Console application workspace.

The command copies from directory `%EUROPLEXUS%\init` (or from `%EUROPLEXUS%\init_quickwin`, in case the `-w` switch is used) to the current directory a Visual Studio .NET 2003 solution named `plex` (consisting of files `plex.sln`, `plex.ncb` and `plex.suo`), and containing the two projects `filter` (sub-directory `filter`, containing file `filter.vcproj`) and `epx` (sub-directory `epx` containing file `epx.vfproj`). If any of these files and sub-directories exists already in the current directory, an error message is issued and the copy actions are not performed.

The projects are pre-set in order to contain the single source file `main.ff` (`main.f` in `epx`). This source file is copied to the current directory from the EUROPLEXUS library, via the `epx_get` command, so as to be sure to get the latest version. In case the `-w` switch is used, the additional files `m_qwin.ff` and `ifwin.ff` are also contained in the project, since these are the only files of EUROPLEXUS that contain QuickWin-related code.

The `debug` and release sub-directories of the project sub-directories are not copied, since they are automatically generated when the solution is re-built. The `main.f` file is also generated automatically from `main.ff` upon re-building.

It is advised to re-build the solution immediately after its generation by the `epx_init` command, i.e. before adding any files into it. To rebuild the solution, open file `plex.sln` by Visual Studio .NET 2003 (simply double-click on it), in the Solution Explorer right-click on "Solution 'plex' (2 projects)" and from the pop-up menu select ***Build Solution***, then click on ***File → Save All***. The following messages typically appear:

```
------ Build started: Project: filter, Configuration: Debug Win32 ------
Performing Custom Build Step
Filtering "d:\users\folco\tmp\""main".ff
1
Filtering done
Build log was saved at "file://d:\users\folco\tmp\filter\Debug\BuildLog.htm"
filter - 0 error(s), 0 warning(s)
------ Build started: Project: epx, Configuration: Debug Win32 ------
Compiling with Intel Fortran 9.0...
..\main.f
Linking...
Build log written to  file://D:\users\folco\tmp\epx\Debug\BuildLog.txt
epx build succeeded.
---------------------- Done ----------------------
    Build: 2 succeeded, 0 failed, 0 skipped
```

The solution is now ready to include new or modified files (as explained in the previous Section) for the development and debugging of EUROPLEXUS.

All script files used in the interactive development and debugging of EUROPLEXUS are listed in the Appendix at the end of the present document. They are also summarized in Table 8.

10.13.22 Bugs and changes of behaviour w/r to previous version

Debugging version

A probable bug has been encountered in the use of the solution as described in the previous Sections. When one launches the executable (from the window-based environment) by clicking on menu ***Debug → Start*** (or by pressing F5) or by clicking on menu ***Debug → Start Without Debugging*** (or by pressing CTRL-F5), the application issues a message window warning that "the epx project configuration is out of date" and asking whether one wants to re-build it, even though one has just re-built it! If you are sure that the configuration is up to date, just click ***No*** on this message box , so the execution starts immediately, else you will have to wait for complete re-build.

Next, the build of the executable seems substantially slower than with the previous version of Developer Studio, especially for the Debug configuration.

Finally, since the organization of projects is different from the previous implementation, in that each project (`filter` and `epx`) resides in a separate sub-directory rather than in the parent directory, when one wants to run a test case the corresponding input file (`*.epx`) must be placed in the `epx` sub-directory, and not in the parent directory. An alternative is to place the input file in the parent directory but then, when the code starts and asks for the input file name, type `..\*.epx` insytead of just `*.epx`.

Checking the arguments in routine and function calls

An important change in behaviour of the compiler was detected. While the previous compiler (Compaq 6.5) automatically and by default checked all arguments in subroutine and function calls, i.e. both their number, nature and position, the new Intel 9.0 compiler does not perform these checks automatically.

To activate these checks, it was found out that one should:

- Compile by using the `/gen-interfaces` option;
- Add also the `/warn:interfaces` option, otherwise any detected mismatches are not reported!

However, this solution to the problem does not seem to be very practical, for the foillowing reasons:

- Compilation with the above flags produces two additional files for each given source myfile.f: a file `myfile.f90`, containing a Fortran 90 `MODULE` encapsulating a set of `INTERFACE` … `END INTERFACE` declaration blocks (one for each subroutine or function contained in the source file) plus the corresponding `myfile.mod` module object file. The number of files is thus greatly increased.
- This of course applies only to files compiled in the current directory. Routines in the library which are used at linkage time are therefore not checked!
- When using the command-line compilation and link with the extra flags, the additional files are taken into account automatically. However, when using the window-based environment for debugging, these extra files have to be manually added to the solution, else they are ignored!

For these reasons, it has been decided (as a temporary solution) to refrain from checking arguments on those installation that use the new compiler. An installation with the previous compiler is maintained, on which the whole code may be recompiled at regular intervals (but not of course at every evolution), just to check that there are no unmatching call arguments.

10.13.23 Simplification of procedures installation and sharing

With complete re-installation of the mirror site the opportunity for simplifying the installation and enhancing the portability of the procedures was taken.

So far, several procedures and other pieces of software contained hard-coded some absoute paths, sometimes including the name of the machine hosting the EUROPLEXUS mirror site. Therefore, each time the machine name or path changed, the procedure had to be updated. The same occurred when using the procedures on a different machine.

To avoid these problems, use is now made of environment variables. These are summarized in Table 10 below.

Table 10 – Summary of environment variables

Variable name	Value (on the mirror site machine)	Meaning

Variable name	Value (on the mirror site machine)	Meaning
EUROPLEXUS	E:\EUROPLEXUS	Directory where the mirror site implementation resides
RUNDIR	\\SM61\USERS\FOLCO\RUN	Directory where Folco's commands (batch files, perl procedures and executables) reside
GNUDIR	\\SM61\USERS\FOLCO\RUN\GNU	Directory where UnxUtils (ports of some common GNU utilities to Win32) commands reside
UTILDIR	\\SM61\E\EUROPLEXUS\UTIL	Directory where the EUROPLEXUS utilities (batch and perl files, executables) reside
NETPBMDIR	C:\NETPBM\BIN	Directory where the NETPBM utilities (mostly executables) for graphical file manipulation reside
GSDIR	C:\GS\GS8.51\BIN	Directory where the GhostScript executable for PostScript file manipulation reside
PATH	…;%RUNDIR%;%GNUDIR%;%UTILDIR%	Path for commands search. Add the three indicated directories towards the end of the existing PATH definition.

The first environment variable (EUROPLEXUS) listed in the Table was already part of the previous implementations, while all others are new (except PATH).

The setting of the first variable deserves some further comments. In the previous implementations this variable was set to E:\EUROPLEXUS on the machine hosting the mirror site, and to \\SM61\E\EUROPLEXUS on other developer's machines on our intranet. Since the latter form is more general, an attempt was made to use it also for the mirror machine.

However, this creates problems in the evolution procedure epx_evol_start. This procedure must work on the evolution directory (on the mirror machine), but this must apparently be a local one, otherwise the following error messages are issued:

```
'\\SM61\E\EUROPLEXUS\Fromcentral'
CMD.EXE was started with the above path as the current directory.
UNC paths are not supported.  Defaulting to Windows directory.
```

To avoid this error, it is suggested to leave the old setting: E:\EUROPLEXUS on the machine hosting the mirror site, and \\SM61\E\EUROPLEXUS on other developer's machines on our intranet. The problem is not relevant to other developers since the epx_evol_start procedure should be launched exclusively from the mirror machine.

The above listed environment variables are now consistently used in all scripts and within the Visual Studion .NET 2003 projects.

To set these variables, proceed as follows:

- Login as administrator.
- Select ***Start → Control Panel → System → Advanced → Environment Variables***.
- Under "System Variables", define or modify the variables as listed in Table 10 (with any appropriate changes in the variable values if you are implementing on a different machine or outside our intranet), then click ***OK*** to apply the changes.

- Reboot the machine. This is necessary so that the newly defined variables are visible within the PATH definition for users different from the administrator.

- Log in as a normal user and in a console window type `set` to check the variable setting. You may also type `set EU` to see all variables whose name starts by `EU`, for example.

10.14 Modifications after August 2005

We describe here the modifications that have been performed after the complete re-implementation of the mirror site of August 2005, decribed in the Fourth Edition of the present document.

10.14.1 Static library for the GLUT package

The Glut package (see Sections 10.10.1 and 10.13.11) used to produce a dynamically linked library (DLL) `Glut32.dll`. It has been decided to produce a static library instead, `Glut32.lib`, for the following reasons:

- Implementation of the code with the DLL required the administrator's priviledge in order to place the DLL file in the system directory. Furthermore, the DLL file had to be distributed in addition to the executable.

- The software house distributing the code has signaled a conflict between OpenGL and their license protection system. It is possible that using a static library this problem is avoided.

The basic Glut distribution contained a Developer Studio project that was used so far to produce the DLL version of the library. To simplify things, a command-line script has been prepared that produces the LIB version instead. This is described in a separate report.

The LIB version (file `Glut32.lib`) has then to be manually copied to the directory:

```
C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\PlatformSDK\Lib
```

The `C:\Windows\System32\Glut32.dll` file might be romoved as far as EUROPLEXUS is concerned, but it is left in place for the moment for compatibility with previous versions of the EUROPLEXUS executable, and also due to the fact that Cast3M still uses it at the moment.

The modifications in the procedures deriving from this change are listed below:

- The linkage procedure `epx_lk` has been modified: the additional library `Glut32.lib` has been inserted between `f90glut.lib` and `bmplib.lib` in the list of the libraries to be used.

- In the .NET solutions for console and QuickWin version the new library `Glut32.lib` has been inserted between `f90glut.lib` and `bmplib.lib` in the 'Properties" of the `epx` project, under ***Linker→ Input→ Additional Dependencies***.

New files produced

As a consequence of using the new library, during the linking process (both when using the `epx_lk` script and via the Visual Studio) two extra files are produced in addition to the `epx.exe` executable:

- `Epx.lib`

- `Epx.exp`

These have to do (apparently) with exported names in the C code of the Glut package, and may be safely ignored by the user. They are not needed to run the resulting EUROPLEXUS executable.

10.14.2 New filtering procedures for the “Fiches”

Two new procedures `stat_anom` and `stat_deve` have been added for the automatic filtering of the “Fiches d’anomalie” and “Fiches de developpement” files, respectively. Note that these require the (local) presence of the “Fiches” files, so they are actually used on the central site (atelier logiciel) and not at the mirror sites. They produce a succint resume of all open and closed fiche files, that will be shortly made available to developers by new buttons in the atelier logiciel.

The syntax reads:

```
stat_anom [-f]

stat_deve [-f]
```

where:

`-f` Option to list just the first line of the fiche “Title” (for the “Fiches de developpemnt”) or “Description” (for the “Fiches d’anomalie”). By default, the entire text is listed, and this normally uses several lines (especially in the case of the “Fiches d’anomalie”).

Each command produces a compact resume (on standard output) of all the currently open fiches, followed by the closed ones. For each fiche, the number, opening date, author and title or description is given. For closed fiches, the closure date and suthor are given as well. The atandard output may be redirected on a file. This facilitates developers e.g. in finding the fiche number related to a certain development or anomaly, in knowing how many (and which) fiches are still currently open, etc.

10.14.3 Full recompilation in case of module evolution

In March 2006 an important modification has been introduced in the procedures for code development. While the use of Fortrant 90 module files in the programming had become more and more frequent, to take full advantage of the features offered by these modern programming entities, the human effort in order to prepare and submit evolution packages to the EUROPLEXUS Development Center had been abnormally increased.

The reason is that when a module is modified (at least in its `PUBLIC` interface), all other program units that use the module (i.e. either directly or indirectly, via other modules) should be re-compiled.

Thus, more and more ofter the developers were obliged to evolve relatively large packages of source files, the vast majority of which contained no actual modifications but had to be recompiled because of changes in a module.

This operation has several drawbacks:

- Each source file to be evolved must be locked via the Web interface to the Developemnt Center.
- A text file containing a description of the changes has to be prepared for each of the evolved files. The text ends up in the history files, which grow up abnmormally thus hiding the relevant information (actual changes).
- Conflicts arise often between the various developers because of routines already locked by another person, which should also be recompiled because of changes in a module.

To avoid all these drawbacks, it was decided to modify the code evolution procedure (at all mirror sites) according to the following rule:

> *Whenever a sources evolution contains at least one module file (i.e. a file with the name* `m_*.ff`*), all sources must be re-compiled.*

In this way, a developer that wants to evolve a module needs not bother about dependencies: all sources will be re-compiled. Therefore, only the source files containing actual modifications need to be transmitted to the Development Center and only such files will be propagated to the mirror sites, thus also drastically reducing the size of the sources evolution packages.

The only disadvantage of the new procedure is that more work is required from the computer. A complete recompilation of the sources takes from a few minutes to some 20 minutes depending on CPU speed. However, this time is small with respect to the CPU needed to pass all the benchmarks. Therefore, the overhead associated with the new evolution strategy is considered acceptable, especially since the code evolution is performed overnight.

The new procedure also reduces the probability of development errors that remain un-detected durinhg evolution and show up only later. Thus the quality assurance of the developments is also increased.

It is worth commenting about local development, i.e. about the procedure to be followed *prior* to submitting an evolution. Performing a full recompilation during the day, as part of the checks to be carried out during normal code development, is clearly too time-consuming. Therefore, developers should not change their habits.

The rules for local development are quite simple.

Whenever a module is modified:

- all dependencies are sought locally (by the command `epx_modtree`, which has been rendered considerably more efficient);
- all the dependent program units are retrieved (by the new command `epx_get_users`);
- all local source files are re-compiled by the command `epx_cmp` (full recompilation is not needed here);
- when submitting files for evolution, make sure to include only those which contain actual modifications.

10.14.4 New FORTRAN Compiler (INTEL 9.1)

A new version of the INTEL Fortran compiler was installed on 20 April 2007:

- Intel Fortran Compiler 9.1.

An attempt to install also a new version of the visual development environment (.NET 2005) resulted in a number of problems, which for the moment have been set aside: the same version of .NET as before (2003) has been retained.

A similar customization of environment variables to that described in Section 10.13.3 for the previous version of the compiler was necessary in order for the compiler to work from the command line.

As administrator, the appropriate environment variables were set permanently as System variables, so that they are visible to all users. In this way the problems are overcome, but the method seems not appropriate e.g. in case of later upgrading of the compilers etc. The setting was done by comparing the output of the set command before and after (interactive) execution of the `ifortvars.bat` script. The result of the `set` command after the patch is listed below, with highlighted the variables which have been added or modified:

```
ALLUSERSPROFILE=C:\Documents and Settings\All Users
APPDATA=C:\Documents and Settings\folco.ELSAUNIT\Application Data
CLASSPATH=C:\Program Files\Java\jre1.5.0_06\lib\ext\QTJava.zip
CommonProgramFiles=C:\Program Files\Common Files
COMPUTERNAME=SM61
ComSpec=C:\WINDOWS\system32\cmd.exe
DevEnvDir=C:\Program Files\Microsoft Visual Studio .NET 2003\Common7\IDE
EDBSUM99=D:\USERS\DBAS\Edbsum99.txt
ESOPE_PARAM=ESOPE=32000000
ESOPE_TEMP=C:\TEMP
EUROPLEXUS=E:\EUROPLEXUS
FP_NO_HOST_CHECK=NO
FrameworkDir=C:\WINDOWS\Microsoft.NET\Framework
FrameworkSDKDir=C:\Program Files\Microsoft Visual Studio .NET 2003\SDK\v1.1
FrameworkVersion=v1.1.4322
```

```
GIBI_FILES=C:\Cast3m\gibi_files
GIBI_START=C:\Cast3m\Bin
GNUDIR=\\SM61\USERS\FOLCO\RUN\GNU
GSDIR=C:\GS\GS8.51\BIN
HOMEDRIVE=C:
HOMEPATH=\Documents and Settings\folco.ELSAUNIT
IDB_PATH=C:\Program Files\Intel\
IFORT_COMPILER90=C:\Program Files\Intel\Compiler\Fortran\9.0
IFORT_COMPILER91=C:\Program Files\Intel\Compiler\Fortran\9.1
INCLUDE=C:\Program Files\Intel\Compiler\Fortran\9.1\Ia32\Include;
        C:\Program Files\Microsoft Visual Studio .NET 2003\VC7\ATLMFC\INCLUDE;
        C:\Program Files\Microsoft Visual Studio .NET 2003\VC7\INCLUDE;
        C:\Program Files\Microsoft Visual Studio .NET 2003\VC7\PlatformSDK\include\prerelease;
        C:\Program Files\Microsoft Visual Studio .NET 2003\VC7\PlatformSDK\include;
        C:\Program Files\Microsoft Visual Studio .NET 2003\SDK\v1.1\include
INTEL_LICENSE_FILE=C:\Program Files\Common Files\Intel\Licenses
INTEL_SHARED=C:\Program Files\Common Files\Intel\Shared Files
LIB=C:\Program Files\Intel\Compiler\Fortran\9.1\Ia32\Lib;
    C:\Program Files\Microsoft Visual Studio .NET 2003\VC7\ATLMFC\LIB;
    C:\Program Files\Microsoft Visual Studio .NET 2003\VC7\LIB;
    C:\Program Files\Microsoft Visual Studio .NET 2003\VC7\PlatformSDK\lib\prerelease;
    C:\Program Files\Microsoft Visual Studio .NET 2003\VC7\PlatformSDK\lib;
    C:\Program Files\Microsoft Visual Studio .NET 2003\SDK\v1.1\lib
LOGONSERVER=\\SM2002
MIF_PATH=C:\Cast3m\gibi_files
MSVCDir=C:\Program Files\Microsoft Visual Studio .NET 2003\VC7
NETPBMDIR=C:\NETPBM\BIN
NUMBER_OF_PROCESSORS=2
OS=Windows_NT
Path=C:\texmf\miktex\bin;
     C:\Perl\bin\;
     C:\Program Files\Intel\Compiler\Fortran\9.1\Ia32\Bin;
     C:\Program Files\Common Files\Intel\Shared Files\Ia32\Bin;
     C:\Program Files\Microsoft Visual Studio .NET 2003\Common7\IDE;
     C:\Program Files\Microsoft Visual Studio .NET 2003\VC7\BIN;
     C:\Program Files\Microsoft Visual Studio .NET 2003\Common7\Tools;
     C:\Program Files\Microsoft Visual Studio .NET 2003\Common7\Tools\bin\prerelease;
     C:\Program Files\Microsoft Visual Studio .NET 2003\Common7\Tools\bin;
     C:\Program Files\Microsoft Visual Studio .NET 2003\SDK\v1.1\bin;
     C:\WINDOWS\Microsoft.NET\Framework\v1.1.4322;
     C:\WINDOWS\system32;
     C:\WINDOWS;
     C:\WINDOWS\System32\Wbem;
     \\SM61\USERS\FOLCO\RUN;
     \\SM61\USERS\FOLCO\RUN\GNU;
     \\SM61\E\EUROPLEXUS\UTIL;
     C:\Vim\Vim63;
     C:\Program Files\QuickTime\QTSystem\;
     C:\Program Files\Intel\IDB\9.1\IA32\Script
PATHEXT=.COM;.EXE;.BAT;.CMD;.PL;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
PROCESSOR_ARCHITECTURE=x86
PROCESSOR_IDENTIFIER=x86 Family 15 Model 2 Stepping 9, GenuineIntel
PROCESSOR_LEVEL=15
PROCESSOR_REVISION=0209
ProgramFiles=C:\Program Files
PROMPT=$P$G
QTJAVA=C:\Program Files\Java\jre1.5.0_06\lib\ext\QTJava.zip
RUNDIR=\\SM61\USERS\FOLCO\RUN
SAMCEFFIELD_EXE=C:\SamcefField\V6.1
SAMTECH_LICENSE_FILE=@sm2002
SAM_WORK=C:\SamWork
SESSIONNAME=Console
SystemDrive=C:
SystemRoot=C:\WINDOWS
TEMP=C:\DOCUME~1\FOLCO~1.ELS\LOCALS~1\Temp
TMP=C:\DOCUME~1\FOLCO~1.ELS\LOCALS~1\Temp
USERDNSDOMAIN=ELSAUNIT.LOCAL
USERDOMAIN=ELSAUNIT
USERNAME=folco
USERPROFILE=C:\Documents and Settings\folco.ELSAUNIT
UTILDIR=\\SM61\E\EUROPLEXUS\UTIL
VCINSTALLDIR=C:\Program Files\Microsoft Visual Studio .NET 2003
VS71COMNTOOLS=C:\Program Files\Microsoft Visual Studio .NET 2003\Common7\Tools\
VSINSTALLDIR=C:\Program Files\Microsoft Visual Studio .NET 2003\Common7\IDE
windir=C:\WINDOWS
```

10.14.5 Customized files for FORTRAN development

Each time a new version of the Fortran compiler is installed, a series of customized files must also be installed at the appropriate locations. These concern graphical developments under OpenGL, Glut and F90GL and are listed below for convenience. Note that they should be copied from the old compiter installation before eventually removing it. The locations listed below may vary across different versions of the compiler and are only indicative.

- In: **C:\Program Files\Intel\Compiler\Fortran\9.0\IA32\Include:**

  opengl_fwrap.mod ← for F90GL
  opengl_gl.mod
  opengl_glinterfaces.mod
  opengl_glu.mod
  opengl_gluinterfaces.mod
  opengl_glut.mod
  opengl_glutinterfaces.mod

opengl_kinds.mod

- In: **C:\Program Files\Intel\Compiler\Fortran\9.0\IA32\Lib:**

  bmplib.lib ← for BMPLIB
  f90GL.lib ← for F90GL
  f90GLU.lib
  f90GLUT.lib

- In: **C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\PlatformSDK\Include\gl:**

  GL.h
  GLAux.h
  GLU.h
  glut.h ← for GLUT
  glutf90.h ← for F90GL

- In: **C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\PlatformSDK\Lib:**

  GlAux.Lib
  GlU32.Lib
  Glut32.lib ← for GLUT (replaces DLL)

- In: **C:\WINDOWS\system32:**

  glu32.dll
  glut32.dll ← no longer used for GLUT (use LIB above)

10.14.6 New strategy for the transfer of evolution packages

The procedure to transfer evolution packages from the central site to the JRC mirror site, described in Section 10.8.2 has been modified because the JRC server machine has been replaced and the new one does *not* allow `ftp` access, i.e. uploading of files from the central mirror site.

To circumvent this problem, the transfer is performed in the following way:

- First, at each evolution the central site prepares a tarred file `europlexus.tar` containing the packages of the last 5 or so evolutions. This file is hosted in a fixed directory of the central mirror site, which may be accessed (but only for downloading) from the outside world.

- Next, the `europlexus.tar` file is copied (downloaded) from the central mirror site to the JRC mirror site (Windows machine inside the firewall). This operation is executed by the Windows machine, which executes a (modified) procedure `epx_ftp_getfiles`. This is a Perl procedure using the `ftp` package `NET::FTP` to access the central mirror site.

- Then, the tar file is unpacked, and any obsolete evolution packages are removed (by comparing their evolution number with the current evolution number). Only the new evolution packages are retained.

- Finally, the evolution may start as usual, by using the files on the mirror site.

The syntax of the command is the same as before:

```
epx_ftp_getfiles
```

and is listed in the Appendix.

For the same reason (no `ftp` available on the new web server), the procedures for updating the executables (`epx_ftp_putexe`) and the user's manuals (`epx_ftp_putman`) have also been modified. The `ftp` command, formerly used by these procedures under Perl, has been replaced by the more secute `pscp.exe` (secure copy) command. This command is part of a small package of free utilities that may be downloaded from the internet, containing also `putty.exe`, which allows to open a `telnet`-like session on a remote machine, under secure shell protocol SSH, and `psftp`, a secure version of `ftp`.

The syntax of the two procedures has remained the same as before (see Section 10.9):

```
epx_ftp_putexe [-w]
```

```
epx_ftp_putman
```

and the updated source is listed in the Appendix.

10.14.7 Optional “patch” switch for the epx_evol_start procedure

Normally the `epx_ftp_getfiles` procedure is called automatically from within the `epx_evol_start` evolution driving procedure. However, when errors occur during an evolution it is sometimes necessary to patch the evolution locally. In this case, the patched evolution package already resides on the local machine (normally under `E:\IIS\Ftp\Europlexus`) so that there is no need (and it woukd be wrong!) to transfer it from the central mirror site.

For this reason, an optional switch `-p` (for “patch”) has been added to the `epx_evol_start procedure`, which allows to bypass the execution of `epx_ftp_getfiles` when the `epx_evol_start` procedure is invoked manually to start a patched evolution.

10.14.8 New command epx_dup_bench

A new command `epx_dup_bench` is added, that duplicates an EUROPLEXUS benchmark (file `.dgibi` plus one or more files `.epx`) under a new name.

The syntax reads:

```
epx_dup_bench oldname<.epx> newname<.epx>
```

where:

`oldname` Name of the benchmark (with or without the `.epx` extension) supposed to be existing on the current directory.

`newname` Name of the new benchmark (with or without the `.epx` extension) supposed not to exist yet on the current directory.

Examples:

```
epx_dup_bench toto02 toto03
```

Duplicates file `toto02.epx` in `toto03.epx` on the current directory, and edits it by changing any occurrence of the string `toto02` into `toto03`. If present, file `toto02.dgibi` is also duplicated (and edited) in `toto03.dgibi`. Any post-treatment files of the form `toto02xxx.epx` are also duplicated and edited onto `toto03xxx.epx`.

10.14.9 Malfunctioning of the AT command

An apparent malfunctioning of the `at` command has been observed during the whole month of April 2007, which prevented the code evolution and backup procedures (`epx_evol_start` and `epx_save`, respectively) to be launche automatically, see Section 6.3.1.

Afrer inspection, it turned out that the problem was only indirectly caused by `at`, and was due to the fact that commands executed via `at` are run under the SYSTEM user. At the beginning of April, the permissions for the entire EUROPLEXUS mirror machine had been reset, in order to accommodate for new developers, and it had been forgotten to activate “full control” for the SYSTEM user.

By correcting the permissions, the `at` command works again as expected. During the debugging process, it has been noted that:

- A log file containing the trace of commands run under at may be found in `C:\Windows\SchedLgU.Txt`. In particular, the return code must be 0 for the command to have run succesfully.

- A new and more powerful version of at (command `schtasks`, see on-line help for a complete syntax and examples) is available, which allows much finer tuning than `at`. However, for the purpose of the EUROPLEXUS evolution, the functionality of `at` is sufficient so the old command is kept for the moment.

10.14.10 Replacing the AT command by SCHTASKS

A problem emerged after the updating of the `epx_ftp_putexe` and `epx_ftp_putman` procedures described in Section 10.14.6. The command `pscp` is used instead of `ftp` for increased security. However, the procedures work when launched from an open session, but fail when started in batch overnight by the `at` command.

The error message is:

```
The server's host key is not cached in the registry. You
have no guarantee that the server is the computer you
think it is.
The server's key fingerprint is:
ssh-rsa 1024 32:c9:86:38:87:cc:ed:07:9a:0f:9d:bf:7c:dc:50:2c
Connection abandoned.
```

The reason seems to be due to the fact that under `at` the command is executed as SYSTEM user, and when a connection to the server is attempted under `at`, the `pscp` utility complains that it has never "seen" this server before. The same connection done from an open session (as folco user) works, because this user has had previous contacts with the server and the information is stored in the registry.

To circumvent the problem, a possible solution is to let the whole evolution procedure (which calls internally `epx_ftp_putexe` and `epx_ftp_putman`) run under a specific user (folco) rather than under the default SYSTEM user. However, this may not be done by the `at` command, but requires the more advanced `schtasks` command.

The syntax by means of `schtasks` is as follows:

```
schtasks /create /tn epx_evo /tr E:\\EUROPLEXUS\\Util\\epx_evol_start.bat /sc daily
/st 01:00:00 /ru folco /rp <folco's password>

schtasks /create /tn epx_sav /tr E:\\EUROPLEXUS\\Util\\epx_save.bat        /sc weekly
/d SUN /st 01:00:00 /ru folco /rp <folco's password>

schtasks /create /tn epx_chk /tr E:\\EUROPLEXUS\\Util\\epx_evol_check.bat /sc daily
/st 06:00:00 /ru folco /rp <folco's password>
```

where `<folco's password>` must be replaced by the currently valid password of the folco user. Two new scripts have been prepared:

`epx_schtasks`

`epx_checkat`

The first script replaces the former `epx_setat` command.

The second script may be used to visualize the scheduled tasks that have been set by the previous command.

To remove the scheduled tasks, use the following commands (by replying "y" to the confirmation question which appears):

```
schtasks /Delete /TN "epx_evo"

schtasks /Delete /TN "epx_sav"
```

```
schtasks /Delete /TN "epx_chk"
```

10.14.11 New Visual Studio (2005) and FORTRAN Compiler (INTEL 10.0)

After the previous installation of the INTEL FORTRAN 9.1 compiler under Visual Studio .NET 2003 (see Section 10.14.4) a serious malfunctioning of the debugger integrated in the MS Visual Studio environment was observed. In particular, exploring complex derived types was not possible or gave erroneous indications. The same problem was reproduced in a very small ad-hoc program of just a few lines, totally unrelated with EUROPLEXUS.

The problem seemed to appear on any machine using the 9.1 version of the compiler (with different minor versions) and the .NET 2003. Recall that these installations required customization of System environment variables to make the compiler work from the command line, although it is not clear whether this is the cause of the observed problems.

To resolve the problems, it was decided to install the newer version of Visual Studio, namely the .NET 2005 version, together with the brand new version of the INTEL Fortran compiler (10.0) on a test machine with a freshly installed copy of Windows XP, at the end of June 2007:

The precise versions of the installed software are as follows:

- MS Windows XP Professional, Service Pack 2;
- MS .NET Framework 1.1
- MS .NET Framework 1.1 Hotfix
- MS .NET Framework 2.0
- MS Visual Studio 2005 Professional Edition
- Intel Fortran Compiler for IA-32 applications, Version 10.0.025
- Intel Fortran Compiler for Intel® 64 applications, Version 10.0.025
- Intel Visual Fortran Compiler 10.0 Integrations in MS Visual Studio

The MS Visual Studio was installed first, followed by the Intel Fortran compiler. After installation, it was verified that the compiler and debugger work on the simple program mentioned above, in visual mode.

10.14.12 Compiling from the command line

To make the compiler work also from the command line, some customizations are necessary, as with previous versions. After installation, an icon opening a console ready for interactive use of the compiler from the command line becomes available. Alternatively, the user may open a normal console window, and then set the environment by executing the `ifortvars.bat` script. This script first calls a second script `vcvars32.bat` that sets the variables for using MS Visual Studio 2005 x86 tools (by calling yet another script called `vsvars32.bat`), and then sets the variables for using the Intel Fortran compiler from the command line. The former settings are necessary in particular when invoking the linkage editor (`link`), while the second ones are necessary for the compilation (`ifort`). Unfortunately, neither of these two techniques is suitable for using the compiler overnight from the automatic evolution procedure, because no console window is opened in that case.

Several solutions to this problem were considered:

1. Setting the environment variables by hand as had been done in previous installations. However, this solution was rejected because it might have been at the origin of the malfunctionings observed with the previous installation (although this is considered unlikely);

2. Doing the same thing but by a script, using the `setx -m` command (see Help on commands). This would be slightly safer and quicker to be applied to multiple machines, but has the same drawback as the first solution.

3. Modifying the compilation command (`epx_cmp`) and the link command (`epx_lk`), so that the necessary variables are set locally before invoking the compiler or linker. This is the preferred solution.

Various ways of implementing the third solution listed above were explored. The first two did not work:

- Calling the `ifortvars.bat` from the `epx_cmp.pl` does not work since apparently the setting of the variables is local to the batch file and is lost after the batch file terminates.

- Setting the variables explicitly (e.g. by `SET toto=tata`) in the PERL file (via the `system` command) does not work when the PERL file is transformed to batch (`.BAT`) by means of the `pl2bat` filter.

A first working solution was found which consists in setting the variables in the PERL file by means of the PERL command:

```
$ENV{'TOTO'} = "tata";
```

This sets the variable for the whole duration of the PERL script (but not permanently) and works after conversion into `.BAT` form by `pl2bat`. The solution is not perfect, but acceptable. The good point is that the environment variables need not be changed after installation since the setting is local, just before invoking the compiler or linker from the command-line procedures. The main drawback is that the two commands (`epx_cmp` and `epx_lk`) must be modified each time a new version of the compiler is installed. This might perhaps be mitigated by adding to the commands a switch allowing to choose the compiler version. Furthermore, the number of variable settings to be added is very large (especially the settings operated by `vcvars32.bat`).

A perhaps better solution (and the one which has been finally retained) is as follows:

- The PERL version of the `epx_cmp` and `epx_lk` scripts is left unchanged;

- The scripts are transformed to `.BAT` files as usual by the `pl2bat` filter;

- Thes resulting batch files are edited by hand and at the beginning is added a call to the `ifortvars.bat` script. It is essential that this call be made in the `.BAT` (header) part of the batch file, and not in the PERL part of the same file, otherwise the setting of variables is not retained.

The structure of the resulting batch file is (conceptually) as follows, with highlighted the portion added by hand:

```
@echo off
call "C:\Program Files\Intel\Compiler\Fortran\10.0.025\IA32\Bin\ifortvars.bat"

@rem = '--*-Perl-*--
@echo off
if "%OS%" == "Windows_NT" goto WinNT
perl -x -S "%0" %1 %2 %3 %4 %5 %6 %7 %8 %9
goto endofperl
:WinNT
perl -x -S %0 %*
if NOT "%COMSPEC%" == "%SystemRoot%\system32\cmd.exe" goto endofperl
if %errorlevel% == 9009 echo You do not have Perl in your PATH.
if errorlevel 1 goto script_failed_so_exit_with_non_zero_val 2>nul
goto endofperl
@rem ';
#!perl
#line 15

. . . (PERL script comes here, omitted for brevity)

__END__
:endofperl
```

The non-highlighted portion is the part added by the `pl2bat` filter in order to wrap-up the original PERL script. Note that the inner batch file is not invoked directly, but via the `call` statement, which

ensures that execution of the parent batch file continues after the child script has terminated (rather than stopping, which would be the default behaviour).

This solution has the following advantages:

- The environment variables need not be permanently modified after installation (or re-installation) of the Visual Studio or of the compiler.
- The original scripts to set the variables are invoked, rather than having to insert the script commands into the local compilation and link procedures. This minimizes the possibility of errors and the changes to be made if the software (e.g. the compiler version) is updated.

To make the implementation slightly more functional, the actual modified batch files read:

```
@echo off
call "epx_setvars.bat"

@rem = '--*-Perl-*--
@echo off
if "%OS%" == "Windows_NT" goto WinNT
perl -x -S "%0" %1 %2 %3 %4 %5 %6 %7 %8 %9
goto endofperl
:WinNT
perl -x -S %0 %*
if NOT "%COMSPEC%" == "%SystemRoot%\system32\cmd.exe" goto endofperl
if %errorlevel% == 9009 echo You do not have Perl in your PATH.
if errorlevel 1 goto script_failed_so_exit_with_non_zero_val 2>nul
goto endofperl
@rem ';
#!perl
#line 15

. . . (PERL script comes here, omitted for brevity)

__END__
:endofperl
```

where `epx_setvars.bat` is a new local batch file, containing the following commands:

```
@echo off

if "%EPX_VARS_SET%" == "OK" goto End

echo Setting the variables ...

rem Modify the following value if the Fortran compiler is upgraded !!!
SET IFV=C:\Program Files\Intel\Compiler\Fortran\10.0.025\IA32\Bin\ifortvars.bat

if exist "%IFV%" call "%IFV%"

SET EPX_VARS_SET=OK

:End
```

This command file uses a new environment variable `EPX_VARS_SET` in order to avoid repeated execution of the variables-setting scripts (which might have the effect of indefinitely increasing the length of the set variables). If the Fortran compiler is upgraded, the only modification should then be in `epx_setvars.bat`, namely at the line which sets the `IFV` variable, as indicated by the comment in the above listed script.

The only caveats, with respect to the previous situation, are:

- If the `epx_cmp.pl` or the `epx_lk.pl` PERL scripts need to be modified, remember to re-apply the customizations listed above (basically the `call` to `epx_setvars.bat`) ***after*** they have been converted by `pl2bat`.
- If the compiler (`ifort`) or the linker (`link`) have to be used ***directly*** in an interactive way (i.e. not by means of `epx_cmp` or of `epx_lk`), remember to set the environment variables first. This may be done either by opening a special console window from the compiler icon, or by invoking `epx_setvars` from a normal console window.

10.14.13 Script to customize files for FORTRAN development

To actually compile EUROPLEXUS, in addition to the above mentioned modifications in the `epx_cmp.bat` and `epx_lk.bat` batch files, it was of course necessary to apply the customizations described in Section 10.14.5, with the following differences:

- Any `9.0` appearing in directory names are changed to `10.0.025`;

- Any `.NET 2003\Vc7` appearing in directory names are changed to `8\Vc`;
- The three header files GL.h, GLAux.h and GLU.h were not copied because they already existed after compiler installation and were identical to the previous installation;
- The two library files GlAux.lib and GlU32.lib were not copied because they already existed after compiler installation and were more recent than in the previous installation;
- The DLL file glu32.dll was not copied because it already existed after compiler installation and was identical to the previous installation.

For the other files, the old version was reused (without recompilation) apparently without problems. A full recompilation will perhaps nevertheless be attempted in the next future to have an up-to-date version of all the libraries.

With these settings, the complete compilation of EUROPLEXUS proceeded without errors.

To speed up this customization on several machines, a script `epx_customize_fortran` has been prepared. This script needs to be edited and adapted each time a new customization becomes necessary, but then it may be used on the various machines.

A sample listing of this script (for copying files from a machine with 9.x compiler to 10.0 compiler) is as follows:

```
$fromdir = '"\\\\sm61\c\Program Files\Intel\Compiler\Fortran\9.0\IA32\Include"';
$todir = '"C:\Program Files\Intel\Compiler\Fortran\10.0.025\IA32\Include"';
system("scopy $fromdir\\opengl_fwrap.mod $todir");
system("scopy $fromdir\\opengl_gl.mod $todir");
system("scopy $fromdir\\opengl_glinterfaces.mod $todir");
system("scopy $fromdir\\opengl_glu.mod $todir");
system("scopy $fromdir\\opengl_gluinterfaces.mod $todir");
system("scopy $fromdir\\opengl_glut.mod $todir");
system("scopy $fromdir\\opengl_glutinterfaces.mod $todir");
system("scopy $fromdir\\opengl_kinds.mod $todir");

$fromdir = '"\\\\sm61\c\Program Files\Intel\Compiler\Fortran\9.0\IA32\Lib"';
$todir = '"C:\Program Files\Intel\Compiler\Fortran\10.0.025\IA32\Lib"';
system("scopy $fromdir\\bmplib.lib $todir");
system("scopy $fromdir\\f90GL.lib $todir");
system("scopy $fromdir\\f90GLU.lib $todir");
system("scopy $fromdir\\f90GLUT.lib $todir");

$fromdir = '"\\\\sm61\c\Program Files\Microsoft Visual Studio .NET 2003\Vc7\PlatformSDK\Include\gl"';
$todir = '"C:\Program Files\Microsoft Visual Studio 8\Vc\PlatformSDK\Include\gl"';
system("scopy $fromdir\\glut.h $todir");
system("scopy $fromdir\\glutf90.h $todir");

$fromdir = '"\\\\sm61\c\Program Files\Microsoft Visual Studio .NET 2003\Vc7\PlatformSDK\Lib"';
$todir = '"C:\Program Files\Microsoft Visual Studio 8\Vc\PlatformSDK\Lib"';
system("scopy $fromdir\\Glut32.lib $todir");
```

10.14.14 Link problems

When linking EUROPLEXUS under the newly installed environment, the following error message is obtained:

```
LINK : fatal error LNK1104: cannot open file 'LIBC.lib'
```

This error is mentioned in the Release Notes of the Intel Visual Fortran Compiler 10.0 for Windows. It is due to the fact that in MS Visual Studio 2005 the static, single-threaded Visual C++ libraries `libc.lib` and `libcd.lib` have been removed. The suggested solution is to specify the threaded and/or DLL forms of the run-time libraries. For example:

- `/libs:static /threads`
- `/libs:dll`
- `/MT`

If nothing is specified, the `ifort` default is to use, with Visual Studio 2005, `/libs:static /threads`, which is the same as `/MT`.

The above reported switches `/libs` etc. are Intel Fortran compiler optional switches that may be specifird after the `/link` switch in the `ifort` command. The compiler calls the Microsoft linker (`link`) and passes the information.

However, in the `epx_lk` command, since August 2005 (see Section 10.13.11) we no longer use `ifort` to link EUROPLEXUS, but rather the `link` command directly. Unfortunately, it is not clear what are the switches to be passed to the linker equivalent to the above compiler switches.

As a temporary workaroud solution, it was attempted to eliminate the error by adding the linker switch:

```
/NODEFAULTLIB:libc.lib
```

within the `epx_lk` command. This apparently solves the problem, since the error message disappears.

It should be noted that, already in the previos version, the above switch was used when building the QuickWin version of the code (`epx_lk -w`). The switch was not used for the standard version because in that case a large number of unresolved references resulted. But with the new environment, it seems that the switch can be used in all cases, because both versions seem to be built up correctly (at least, no error messages occur).

The EUROPLEXUS version produced with the modified link procedure passes all standard non-regression benchmark tests.

10.14.15 Using the visual debugger

To debug visually under MS Visual Studio 2005, the "solution" prepared under .NET 2003 (see Section 10.13.19) has been updated to run under the new environment. By simply opening the old solution (`plex.sln`) with MS Visual Studio 2005, a conversion wizard pops up. After a couple of clicks (accept all the proposed defaults) the conversion is performed, apparently without any errors. The converted solution was saved, to make the changes permanent, and the three modified files (`plex.ncb`, `plex.sln` and `plex.suo`) were replaced in the EUROPLEXUS Init directory.

When building the solution, the following link warning appears:

```
LINK : warning LNK4098: defaultlib 'LIBCMT' conflicts with use
of other libs; use /NODEFAULTLIB:library

Embedding manifest …
```

We tried out the suggested workaround. In the open solution click on the `epx` project to make sure it is selected, then select ***Project → Properties → Linker → Input***. Under ***Ignore Specific Library*** you should already find `libc.lib`. Add also `libcmt.lib` (separated by a blank). Save the project and re-build. Unfortunately, now a fatal error (not a warning) occurs at link time. It is the same error discussed in the previous Section:

```
LINK : fatal error LNK1104: cannot open file 'LIBC.lib'
```

Therefore, for the moment we leave things as they are and do not exclude the `libcmt.lib` library from the project (thus accepting the apperance of the above warning message).

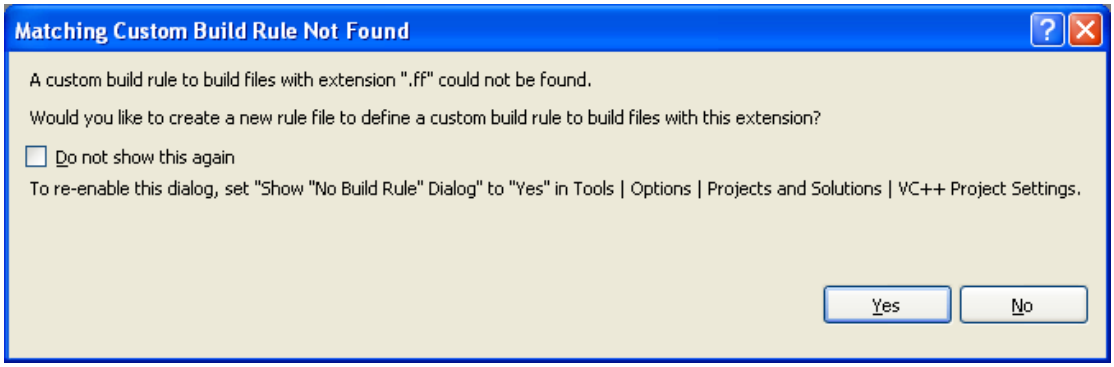
10.14.16 Improvements in the Visual Studio Solution

A couple of improvements were made in the MS Visual Studio solution after converting it to the new format (2005) as explained in the previous Section:

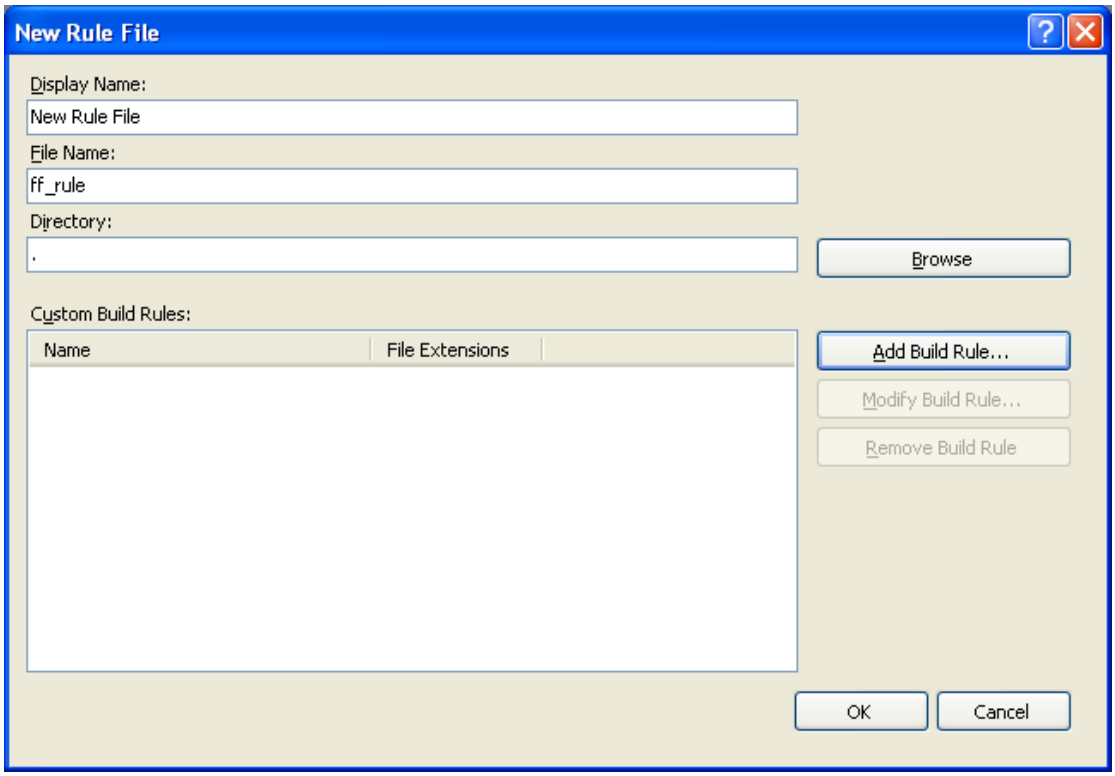
- Debug information for Fortran PARAMETER constants was activated by clicking on the `epx` project to make sure it is selected, then selecting ***Project → Properties → Fortran → Debugging*** and by setting ***Information for PARAMETER Constants*** to ***All*** instead of None (which is the default). This corresponds to command-line compiler switch `/debug-parameters:all`.

- Trapping of unitialized variables was activated by clicking on the `epx` project to make sure it is selected, then selecting ***Project → Properties → Fortran → Data*** and by setting ***Initialize stack values to an unusual value*** to ***Yes*** instead of No (which is the default). This corresponds to command-line compiler switch `/Qtrapuv`.

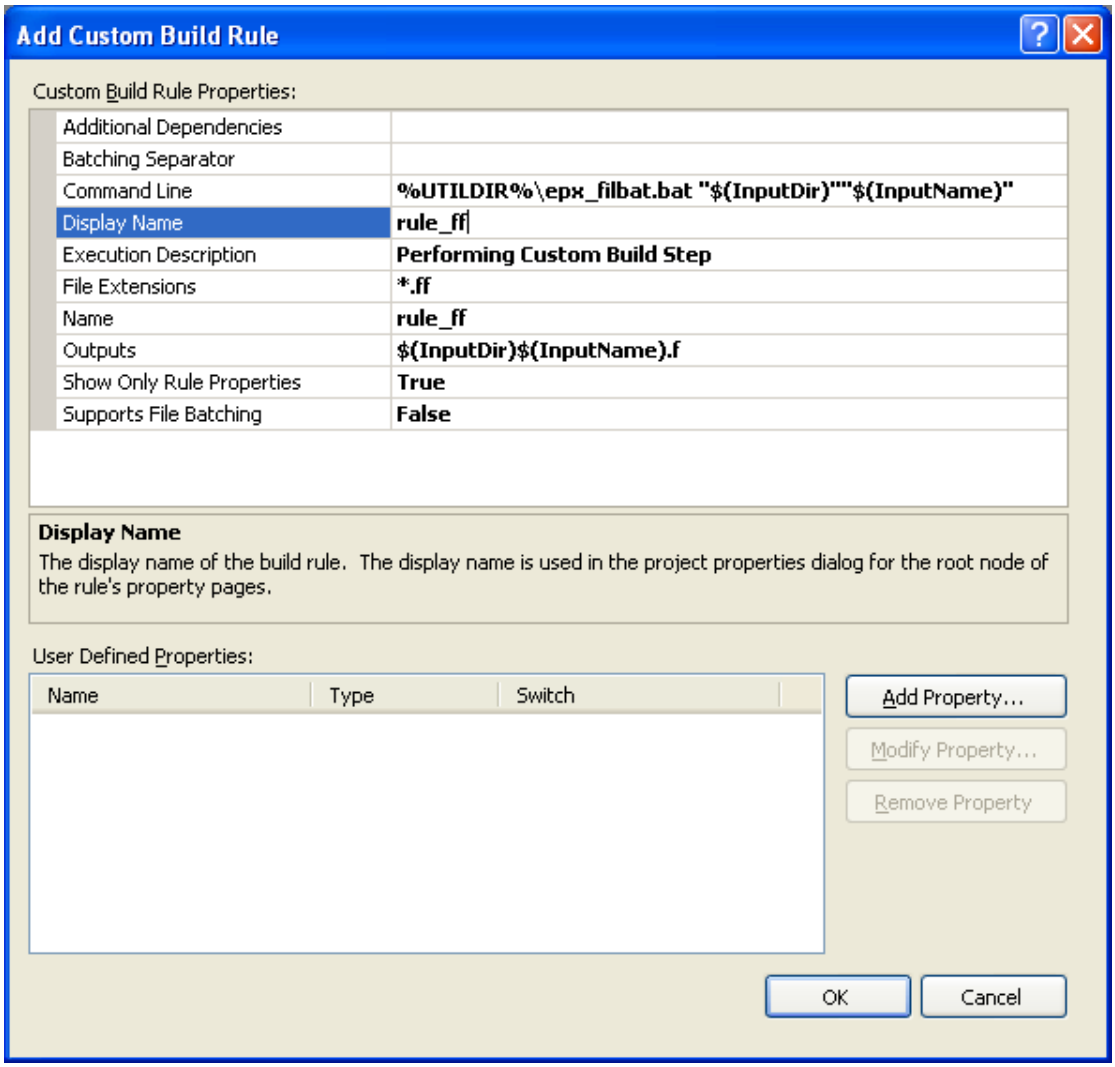
Finally, it was noticed that, when new .ff source files are added to the Filter sub-project (see Section 10.13.20) a pop-up window appears offering the possibility of creating a custom build rule associated with files having that extension. This greatly simplifies the addition of new source files to the solution.



A new Rule was created by clicking on `Yes`. The following dialog appears: in the Directory, choose `.` (the current directory) and as file name enter `ff_rule`, as shown.



Then click on `Add Build Rule` The following Add Custom Build Rule dialog appears (see next Figure). Edit the various fields until the contents is as shown next (by clicking on the various fields some information on the meaning of the various things appears), then click on OK. The new custom build rule is created in file `.\ff_rule.rules` and new files with the extension `.ff` may be added to the filter sub-project without having to manually set their filtering rules bu copying them from `main.ff`, as was the case in the previous installations.



The contents of the `.\ff_rule.rules` file is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<VisualStudioToolFile
  Name="New Rule File"
  Version="8.00"
  >
  <Rules>
    <CustomBuildRule
              Name="rule_ff"
              DisplayName="rule_ff"
              CommandLine="%UTILDIR%\epx_filbat.bat &quot;$(InputDir)&quot;&quot;$(InputName)&quot;"
              Outputs="$(InputDir)$(InputName).f"
              FileExtensions="*.ff"
              ExecutionDescription="Performing Custom Build Step"
              >
              <Properties>
              </Properties>
    </CustomBuildRule>
  </Rules>
</VisualStudioToolFile>
```

An attempt to put the Custom Rule file in a fixed directory (say e.g. `E:\EUROPLEXUS\Rules`) failed because, when the solution is copied frtom the `Init` directory to the place when one wants to do thr debugging, upon opening the solution an error occurs because the Rule file is apparently sought in a relative directory to the place where it has been first created.

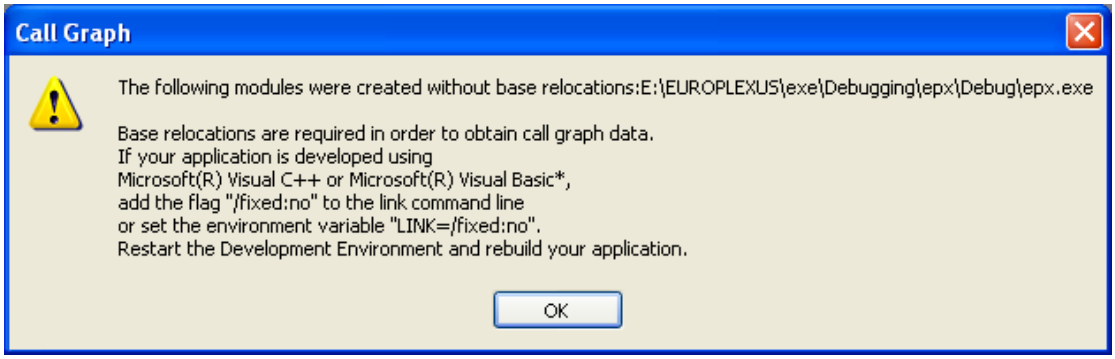
As a consequence of this behaviour, the rule file must be copied to the current directory along with the other solution files. This action is therefore added to the `epx_init` command so that the user does not have to bother with this.

10.14.17 Profiling with Intel VTune

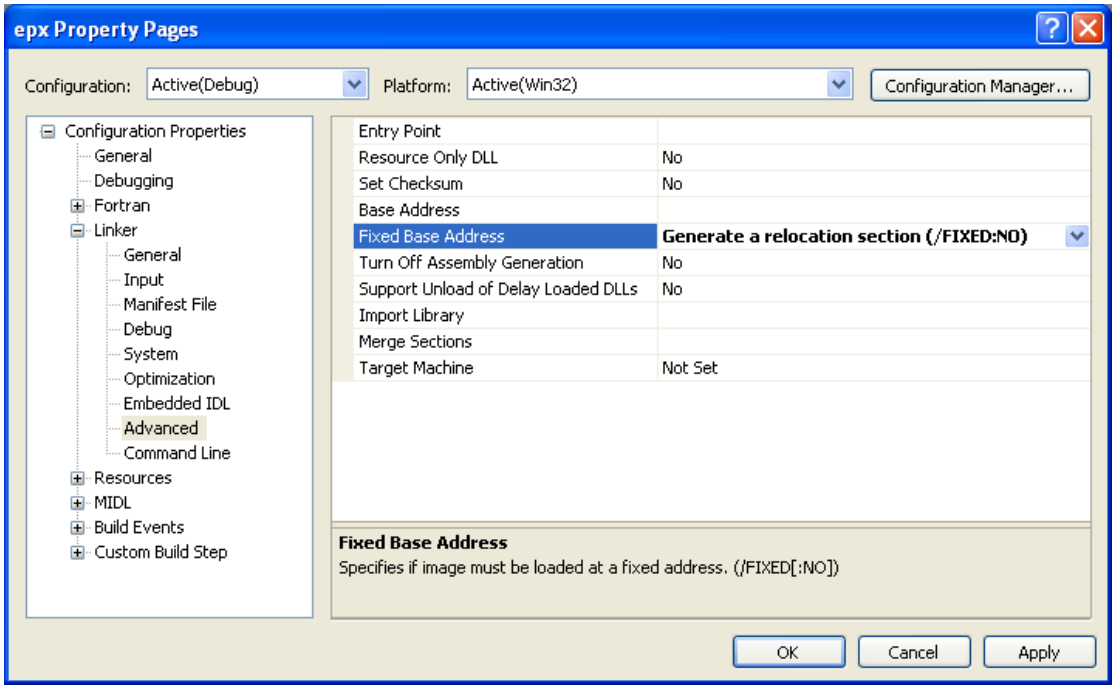
For profiling purposes, the Intel VTune Performance Analyzer was installed, precisely Version 9.0, Build 24052. The installer proposes also the installation of the Intel Tread Profiler 3.1, which was

accepted. VTune is integrated with the MS Visual Studio environment if so chosen during the installation. In this way, it may be started directly from Visual Studio rather than as a separate product. A new Tab named ***Tuning Browser*** appears, next to the ***Solution Explorer*** Tab. By opening the Tuning Browser, the two sub-projects (`epx` and `filter`) appear. An *activity* may be added by right-clicking on the `epx` project and then selecting ***New Activity***. A wizard appears that guides you through the process.

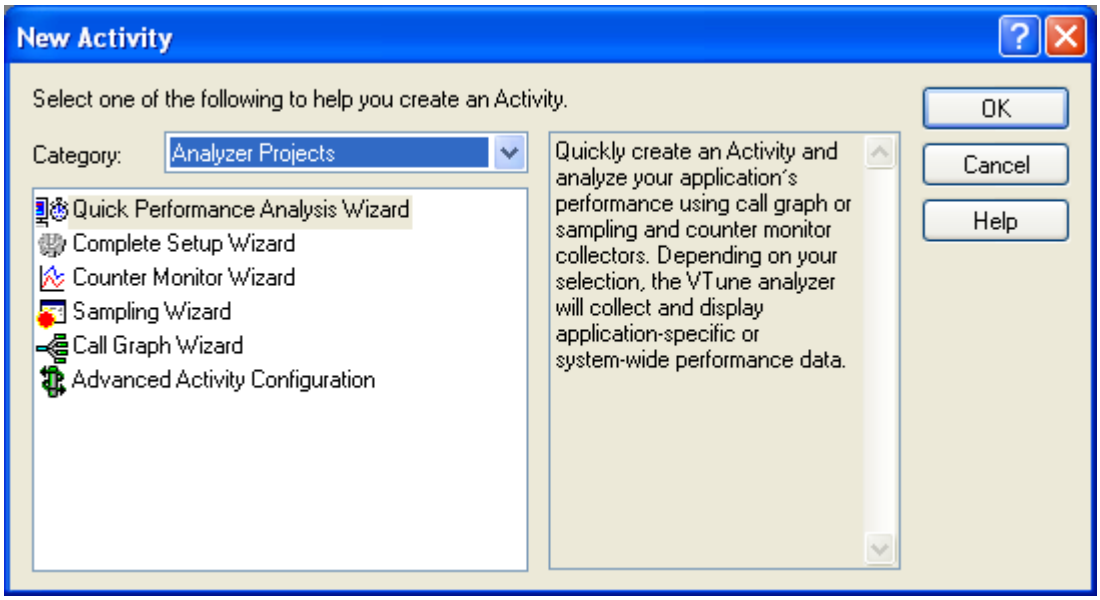
Three types of Data Collecting activities are possible: **Sampling**, **Counter Monitor** and **Call Graph**. The latter seems most interesting, but it is only possible if the executable module has been produced with base relocation, as appears from the following message.



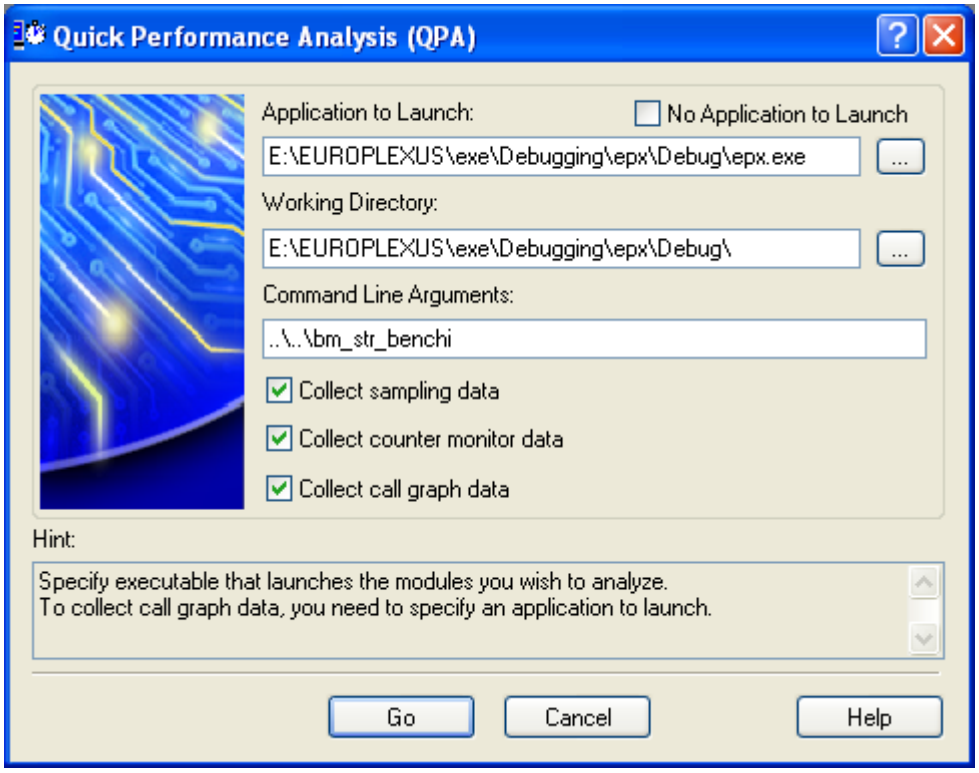
To activate base relocation in `epx`, click on the `epx` project to make sure it is selected, then select ***Project → Properties → Linker → Advanced*** and set ***Fixed Base Address*** to ***Generate a relocation section***. This corresponds to command-line linker switch `/fixed:no`.



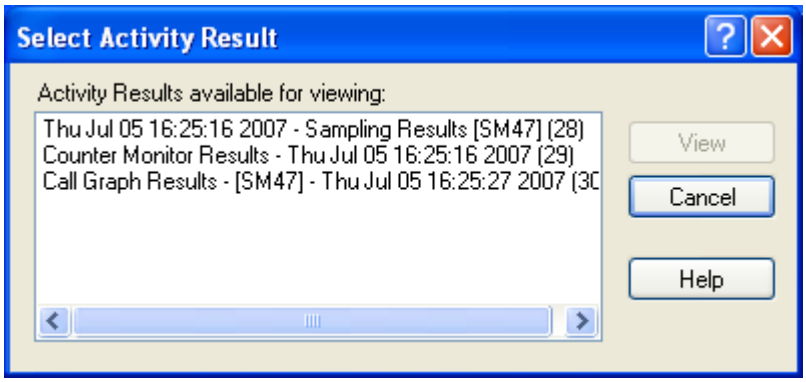
To start a tuning activity, right-click on the `epx` project and then select ***New Activity***. The following wizard appears. Select ***Quick Performance Analysis Wizard*** and click on OK.



Fill in the window that appears (see below) with the relevant ***Application to launch*** (`epx.exe`) and ***Command Line Arguments*** (the name of the input file for EUROPLEXUS). Note that the input file name must be preceeded by `..\..\` to navigate up two directories from the location where VTune operates (i.e the ***Working Directory*** appearing just above). By default, only the ***Collect sampling data*** box is checked. You may want to check also the other two boxes (***Collect counter monitor data*** and ***Collect call graph data***), as shown below, for full collecting.



By clicking on the Go button, the collecting activity starts. VTune automatically runs the code twice and collects the performance data. If the first execution is long, the tuner will probably try to stop it after a few seconds. In the Output pane, messages appear that let you follow the status of the data collecting activity. At the end, you should obtain the following window poppoed up:



Click on one of the available results to see it displayed in VTune.

10.14.18 Updating the EUROPLEXUS accessory libraries

An update of the EUROPLEXUS accessory libraries is performed, following the installation of the new development environment described in Section 0.

Libplex_c library

First of all, the EUROPLEXUS C library `libplex_c.lib` is ***no longer used***, following recent changes in the source. Therefore, this library is removed from the link procedure (`epx_lk.pl`) and also from the debugging solution.

BLAS library

The BLAS library was re-generated by the following commands:

```
IFORT /c /optimize:5 libblas.f
LIB /OUT:Libblas.lib libblas.obj
mv Libblas.lib E:\EUROPLEXUS\Library
```

SPLIB library

The SPLIB library was re-generated by the following commands:

```
IFORT /c /optimize:5 *.f
LIB /OUT:Libsp.lib *.obj
```
mv Libsp.lib E:\EUROPLEXUS\Library

GLUT library

It was checked that the GLUT library is still currently in the version 3.7.6 (8 November 2001) as previously, consisting in the file `glut-3.7.6-src.zip`. To the source, various modifications as described in reference 12 have to be applied, which are contained in the package `Glut_f90gl_modifs.tar.gz` by the author.

Recall that recently the EUROPLEXUS mirror site has been modified so as to use the static version of the Glut library rather than the dynamic one, in order to simplify the code installation process.

In order to actually build up and install the static library version of the Glut package (file `Glut32.lib`), one should proceed as follows.

First, extend the GLUT package with the functionality needed in EUROPLEXUS. To this end, the following files need be modified:

- `glut-3.7.6\include\GL\glut.h`
- `glut-3.7.6\include\GL\glutf90.h`
- `glut-3.7.6\lib\fglut\fglut.c`

- `glut-3.7.6\lib\glut\include\GL\glutf90.h`
- `glut-3.7.6\lib\glut\glut.def`
- `glut-3.7.6\lib\glut\glut_event.c`
- `glut-3.7.6\lib\glut\glut_fcb.c`
- `glut-3.7.6\lib\glut\glut_win.c`
- `glut-3.7.6\lib\glut\glutint.h`
- `glut-3.7.6\lib\glut\win32_winproc.c`

To this end, simply copy the above listed modified files from the package `Glut_f90gl_modifs.tar.gz` by the author onto the standard GLUT package. Then:

- Create a directory, say `Glut`, under `E:\EUROPLEXUS`.
- Copy into this directory all the C source files from the (modified) Glut distribution, namely the files `glut-3.7.6\lib\glut\*.c`, with the following exceptions: files `capturexfont.c`, `glut_menu.c`, `glut_menu2.c` and `layerutil.c` (these files are unused).
- Copy into this directory all the C header files from the Glut distribution, namely the files `glut-3.7.6\lib\glut\*.h` (files `glutbitmap.h`, `glutint.h`, `glutstroke.h`, `glutwin32.h`, `layerutil.h`, `stroke.h`, `win32_glx.h` and `win32_x11.h`).
- Copy into this directory all the C header files `glut-3.7.6\include\GL\glut.h` and `glut-3.7.6\include\GL\glutf90.h`.
- Compile all files and produce the library `Glut32.lib`, e.g. by the following perl script (`make_all.pl`):

```
system ("del *.obj");
system ("del Glut32.lib");
system ("cl -c -o2 *.c");
system ("lib /OUT:Glut32.lib *.obj");
system ("del *.obj");
exit;
```

- Install the library by manually copying the file `Glut32.lib` to the directory:

```
C:\Program Files\Microsoft Visual Studio 8\Vc\PlatformSDK\Lib
```

Finally, do not forget to copy *by hand* the file

```
glut-3.7.6\include\GL\glutf90.h
```

into the directory

```
C:\Program Files\Microsoft Visual Studio 8\Vc\PlatformSDK\include\GL
```

F90gl libraries

Modifications refer to the source of the latest release of F90GL, which for the Windows platform is version 1.2.8 (note that the one for Unix is 1.2.10, but that version has no changes that affect the Windows platform with respect to 1.2.8). Obtain the file `f90gl128.zip` from the address http://math.nist.gov/f90gl/software.html.

Unpack the zipped file in a directory, say `E:\EUROPLEXUS\F90gl`: this will produce a directory `f90gl-1.2.8` with various files and sub-directories in it. The following source files need be modified:

- `f90gl\glut\cwrapglt.c`
- `f90gl\glut\fwrapglt.fpp`
- `f90gl\glut\intrfglt.fpp`

Simply replace them with the modified versions from package `Glut_f90gl_modifs.tar.gz` by the author.

The main directory contains makefiles for the various supported platforms. The one for MS Windows and the Intel compiler is the batch file `mf8nio.bat` Note that the files with an extension `.fpp` are Fortran 90 files that need to be processed by a pre-processor (`sppr.exe`) that comes with the standard `f90gl` distribution (see under `f90gl\util`).

This script (in directory `f90gl-1.2.8`) has to be modified as follows:

- change the value of the `WININC` variable from: `set WININC=F:\Program Files\ Microsoft SDK\include` to: `set WININC=c:\Program Files\Microsoft Visual Studio 8\Vc\PlatformSDK\Include`.

Then in order to compile and build up the standard version, just type:

```
mf8nio
```

This will produce the following library files:

- `f90gl-1.2.8\lib\f90GL.lib`
- `f90gl-1.2.8\lib\f90GLU.lib`
- `f90gl-1.2.8\lib\f90GLUT.lib`

Copy these files to the Intel compiler's `IA32\LIB` directory (on the author's machine, this is `C:\Program Files\ Intel\Compiler\Fortran\10.0.025\IA32\Lib`).

The following eight Fortran module files are also produced:

- `f90gl-1.2.8\lib\opengl_fwrap.mod`
- `f90gl-1.2.8\lib\opengl_gl.mod`
- `f90gl-1.2.8\lib\opengl_glinterfaces.mod`
- `f90gl-1.2.8\lib\opengl_glu.mod`
- `f90gl-1.2.8\lib\opengl_gluinterfaces.mod`
- `f90gl-1.2.8\lib\opengl_glut.mod`
- `f90gl-1.2.8\lib\opengl_glutinterfaces.mod`
- `f90gl-1.2.8\lib\opengl_kinds.mod`

Copy these files to the Intel compiler's `IA32\Include` directory (on the author's machine, this is `C:\Program Files\ Intel\Compiler\Fortran\10.0.025\IA32\Include`).

If during the execution of the `mf8nio` script you get compilation errors (e.g. in `cwrapglt.c`), make sure you have copied the `glutf90.h` include file as explained in the previous Section.

Bmplib library

The library was re-built by copying the previous solution to a directory `E:\EUROPLEXUS\Bmplib`, opening the solution (which converts it automatically) and building it. At the end, the library file `bmplib.lib` is copied by hand to the directory `C:\Program Files\ Intel\Compiler\ Fortran\10.0.025\IA32\Lib`).

10.14.19 Problem in the debug version with OpenGL

Following the recent changes described in the previous Sections, a problem arose when using the debug version of the code in conjunction with OpenGL graphics rendering. The code stops at run time with the following message, apparently due to an invalid floating point number:

```
First-chance exception at 0x00c15fe5 in epx.exe: 0xC00002B5:  Multiple floating point traps.
First-chance exception at 0x00c79fc8 in epx.exe: 0xC0000091: Floating-point overflow.
epx.exe has triggered a breakpoint
```

Inspection of the call stack reveals that the problem arises when calling the `glutInit` initialization function for the OpenGL rendering, and more precisely when this function calls an initialization function for the time (`gettimeofday`):

```
   epx.exe!_gettimeofday()  + 0x68 bytesC
   epx.exe!___glutInitTime()  + 0x18 bytes      C
   epx.exe!_glutInit@8()  + 0x4ba bytes C
   epx.exe!___glutInitWithExit@12()  + 0x18 bytes      C
   epx.exe!_FGLUTINIT()  + 0x65 bytes   C
   epx.exe!_FGLUTINIT()  + 0x4c bytes   C
   epx.exe!_OPENGL_GLUT_mp_F9XGLUTINIT()  + 0x324 bytes
   epx.exe!_M_RENDER_mp_RENDER()  + 0x889 bytes
   epx.exe!_TKPRIN()  + 0x415 bytes
   epx.exe!_IMPSOR()  + 0x2a7e bytes
   epx.exe!_CALCUL()  + 0x610 bytes
   epx.exe!_PRINC()  + 0x1f48 bytes
>  epx.exe!EUROPLEXUS()  Line 66 + 0x66 bytes   Fortran
   epx.exe!_main()  + 0x65 bytes
```

Tests show that:

- The same behaviour occurs with the code compiled and linked from the command line (`epx_cmp`), while the version compiled with optimization (`epx_cmp -o`) works fine. The error message from the command-line version is:

  ```
  forrtl: error(65): floating invalid
  ```

- The link command actually used (with or without `-o`) has no effect.

- This occurs even though the only subroutine compiled in the project (or from the command line) is the main program (`main.ff`) while all the remaining parts of the code are taken from the libraries.

The error occurs in the Glut library, which has been recently recompiled without modification (see Section 10.14.18). By restoring the old version of the library (`glut32.lib`), the error disappears. The version of the f90gl libraries (`f90gl.lib`, `f90glu.lib`, `f90glut.lib`), whether old or new, has no effect.

An attempt to set up a complete MS Visual Studio solution including also the f90gl and glut sources for debugging seems quite complex (although it would be certainly worthwhile) because of the mixture of Fortran and C parts of code.

Rather than using the whole EUROPLEXUS project, tests are continued on the simple example used in the report describing the modifications in the Glut package, see reference [12]. Two versions of the example exist: one is purely C (`test.c`), the other is a mixture of Fortran and C via f90gl (`test.f`, `m_data.f`, `m_callbacks.f`).

The C version of the test works without errors, whether compiled from the command line:

```
cl test.c kernel32.lib user32.lib gdi32.lib
```

or built as a debuggable MS Visual Studio solution. It is interesting to note, incidentally, that there is no need to specify the Glut library (`glut32.lib`) nor by the way the OpenGL libraries (`opengl32.lib`, `glu32.lib`) in the above build command line, while the other listed libraries are necessary to resolve all the symbols. Inspection reveals that this is due to the fact that `test.c` includes the header `glut.h`, which contains the following pragma instructions, telling the linker explicitely to search for these libraries:

```
# pragma comment (lib, "opengl32.lib")
# pragma comment (lib, "glu32.lib")
# pragma comment (lib, "glut32.lib")
```

Some experimentation with the Fortran version of the example finally reveals that the problem is due to the presence of the `/Qtrapuv` flag in the generation of the debug version. This flag had been introduced in Section 10.14.16 to activate trapping of unitialized variables by clicking on the `epx` project to make sure it is selected, then selecting ***Project → Properties → Fortran → Data*** and by setting ***Initialize stack values to an unusual value*** to ***Yes*** instead of No (which is the default). This corresponds to command-line compiler switch `/Qtrapuv`.

By removing this switch, the debuggable version of both the simple example and of the whole EUROPLEXUS code runs without errors in the OpenGL part. Unfortunately, it is not simple to find what causes the error when the switch is present: most probably it must be some uninitialized variable in the Glut part. This would require debugging the complete (Fortran and C) version, which is not feasible at the moment (but would indeed be quite interesting).

As a workaround solution, it is therefore decided to permanently remove the `/Qtrapuv` flag both from the MS Visual Studio solution, debug configuration, and from the command-line compilation script `epx_cmp`.

10.14.20 Migration and re-organization of the EUROPLEXUS mirror site

In September 2007 it is decided to migrate the EUROPLEXUS mirror site to a dedicated machine, for safety reasons. The host thus changes from `sm61` to `sm47`. The new host machine contains only the minimal software required for the evolution of the EUROPLEXUS package. The first evolution performed on the new host is #1361, on 27 September 2007.

Environment variables

The following environment variables need to be changed:

Variable	Old value	New value
GNUDIR	\\SM61\USERS\FOLCO\RUN\GNU	\\SM47\E\EUROPLEXUS\RUN\GNU
RUNDIR	\\SM61\USERS\FOLCO\RUN	\\SM47\E\EUROPLEXUS\RUN
UTILDIR	\\SM61\E\EUROPLEXUS\UTIL	\\SM47\E\EUROPLEXUS\UTIL
PATH	Above three directories	Above three directories

Note that the `RUN` and `RUN\GNU` directories (containing several executables), that were previously located unde a User's directory (`\USERS\FOLCO`) are moved under the `EUROPLEXUS` directory for consistency.

New subdirectories

Two additional subdirectories are created under `EUROPLEXUS`:

- The first one is called `PROC` and contains the Cast3m procedures previously located under an old Plexis-3C directory.

- The other one is called `FTP` and replaces the former `E:\IIS\ftp\EUROPLEXUS` directory that hosts the evolution files transferred by the `epx_ftp_getfiles` procedure. In fact, the IIS services are no longer needed and therefore are not installed any more. The two procedures `epx_evol_start` and `epx_ftp_getfiles` are modified to account for the new directory.

Automaticupdate of executables and manuals on the web site

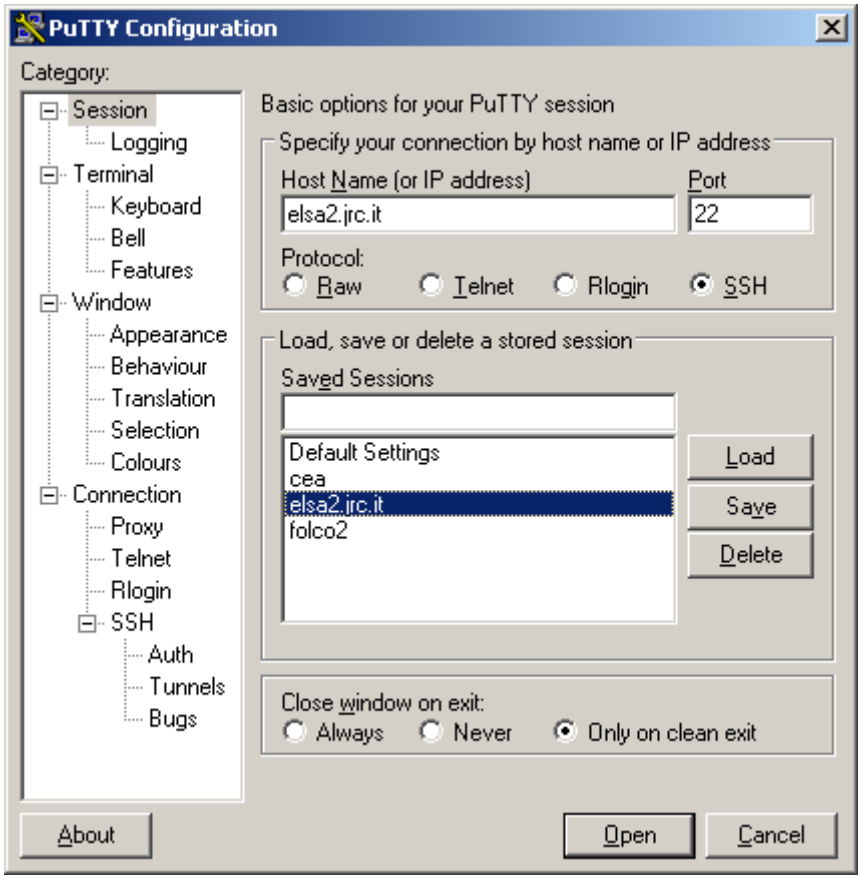
The same problem emerged as mentioned in Section 10.14.10 and related to the `epx_ftp_putexe` and `epx_ftp_putman` procedures for the automatize updating of the executables and manuals on the Consortium web site. The error message is:

```
The server's host key is not cached in the registry. You
have no guarantee that the server is the computer you
think it is.
The server's key fingerprint is:
ssh-rsa 1024 32:c9:86:38:87:cc:ed:07:9a:0f:9d:bf:7c:dc:50:2c
Connection abandoned.
```

To solve the problem, it is necessary to start once a communication by hand with the remote computer. **Important note**: to do this, log in as the folco user because the `epx_schtasks` command sets the scheduled tasks to be executed by the folco user (see Section 10.14.10). Logging in as another user, event as administrator, does not help in this case! Once logged in, in a console window, type the command:

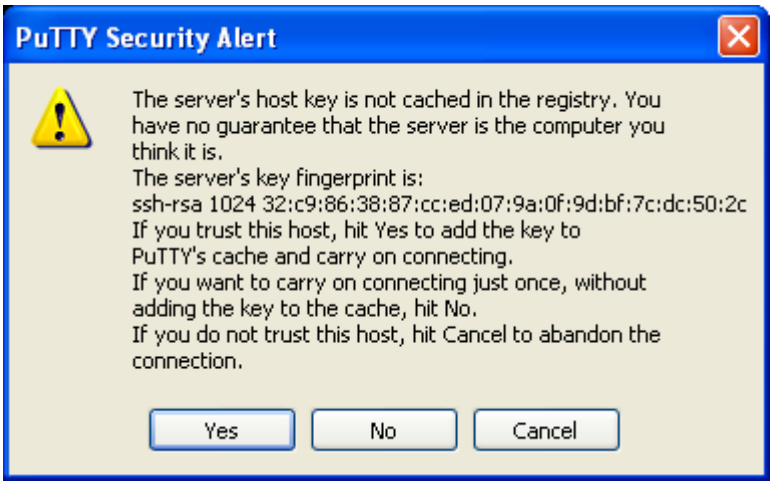
```
putty
```

which opens a dialog window. Check the `SSH` box (secure shell), and as machine address insert `elsa2.jrc.it`:



At this point a furter dialog box should open with a message similar to the above error message (see figure below). Accept the connection and do a login on the remote machine elsa2.jrc.it (e.g. as europlexus user). Then log out and close the connection. From now on the automatic procedures

`epx_ftp_putexe` and `epx_ftp_putman` should work properly when started by the `schtasks` command.



10.14.21 Run-time checking

A very useful feature of the Intel compiler is the ability to check for various things at run time. This includes, in particular, the verification that all local variables are assigned a value before using them, and the verification that local array indexes are within the declared dimension.

To activate these verifications, use is made of the switche `/check:all` during compilation. To use this, the `epx_cmp` compilation procedure is modified by adding a new switch `-c`:

```
epx_cmp [-q] [-o] [-w] [-c] [-k key] [-x] [-g] [file(s)[.ff]]
```

where:

`-c` Option to compile with full run-time checks, i.e. by using the switches `/check:all`. To be used normally without the `-o` switch.

To check a specific routine, the advised commands are as follows:

- Compile all local sources with `-c` and without `-o`:

```
epx_cmp -c
```

- Link without `-o`:

```
epx_lk
```

Upon running the program normally, if a run-time error occurs in the compiled routines, a full traceback should appear.

The `epx_lk` link procedure is also modified by adding a new switch `-c`:

```
epx_lk [-l|-L] [-o] [-w] [-c] [-f] [-p] [-e name] [-s size]
```

where:

`-c` Option to link with library generated by using the switch `/check:all`, which allows checking the entire code.

To check a specific routine, the advised commands are as follows:

- Compile all local sources with `-c` and without `-o`:

```
epx_cmp -c
```

- Link with `-c` and without `-o`:

```
epx_lk -c
```

10.14.22 Updating PERL to solve problem with automatic e-mail

Starting from February 2008 a problem arose with the automatic e-mail sent at each evolution from the JRC mirror site. The mail did not work any more because of changes (enhanced security) in the JRC mailing policy.

The error message concerned the lack of an AUTH command.

To solve the problem, the following steps were taken:

- The newest version of the PERL package was installed. This is Version 5.10.0 (Build 1002) and replaces version 5.8.7 (Build 813).
- The newest version of the SendMail package was installed. This is version 2.09 and replaces the previous version 0.78. The new package was downloaded from: http://www.tneoh.zoneit.com/perl/SendMail.
- The `epx_mail` command was changed in order to make it conforming with the new SendMail package.

The new `epx_mail` script is listed in the Appendix. The functionality and syntax is unchanged with respect to the previous version. The only difference is that from now on the evolution log file is sent as an attachment (text file) rather than being embedded in the message body.

The new Perl package no longer has the conflicting `HEAD.BAT` command as the previous version. However, the `Date.pm` package must still be edited under C:\Perl\Site\Lib\HTTP in order to export the `time2iso` function (Just create the C:\Perl\Site\Lib\HTTP folder anew and copy therein the `Date_ori.pm` and `Date.pm` packages from the previous Perl version).

10.14.23 New BLAS and Lapack libraries

In October 2008 a new version of the BLAS library was installed, and the Lapack library was added. The reson was the fact that it was noticed that the EUROPLEXUS routines for the calculation of eigenvalues and eigenvectors failed in some cases, especially in the presence of multiple values.

The Lapack software (Version 3.1.1) was downloaded and installed. This includes a (newer) version of the BLAS software as well, which replaced the previous BLAS library. The `epx_lk` linking command had to updated, in order to include the new library (Lapack).

Here are the installation notes:

How to prepare BLAS/Lapack libraries for EUROPLEXUS:

- Get file `lapack-lite-3.1.1.tgz` from the Lapack web site: `http://www.netlib.org/lapack/`. This comes down as `lapack-lite-3.1.1.gz`!
- Uncompress it:

```
gunzip lapack-lite-3.1.1.gz
```

- This generates `lapack-lite.3.1.1`.
- Rename it (else it won't be possible to extract the files by `tar`):

```
mv lapack-lite.3.1.1 lapack-lite.3.1.1.tar
```

- Now extract the files:

```
tar xvf lapack-lite.3.1.1.tar
```

- Create directory for Lapack sources:

```
mkdir Liblapack
```

- Copy all Lapack sources to this directory:

```
cp lapack-lite.3.1.1\SRC\*.f Liblapack
```

- Add to this directory the following two sources from the Lapack `INSTALL` directory:

```
cp lapack-lite.3.1.1\INSTALL\dlamch.f Liblapack

cp lapack-lite.3.1.1\INSTALL\slamch.f Liblapack
```

- NOTE: the `lsame.f` source is NOT added because it is identical (apart some blanks and formatting) to the lsame.f contained in the BLAS distribution to be built next.

- Remove from this directory the file xerbla.f, because it is identical (apart some blanks and formatting) to the xerbla.f contained in the BLAS distribution to be built next.

```
rm Liblapack\xerbla.f
```

- Compile and generate library:

```
epx_setvars

ifort -c /optimize:5 Liblapack\*.f

lib /OUT:Liblapack.lib *.obj

del *.obj
```

- This generates Liblapack.lib (~ 6.5 MB).

- Create directory for BLAS sources:

```
mkdir Libblas
```

- Add to this directory all the BLAS sources which come with this distribution package (including `lsame.f` and `xerbla.f`):

```
cp lapack-lite.3.1.1\BLAS\SRC\*.f Libblas
```

- Compile and generate library:

```
epx_setvars

ifort -c /optimize:5 Libblas\*.f

lib /OUT:Libblas.lib *.obj

del *.obj
```

- This generates Libblas.lib (~ 0.4 MB).

10.14.24 Ad-hoc evolution procedure for the 64-bit version

An *ad-hoc* evolution procedure has been prepared for the Windows 64-bit version, command `epx_evol_64`. The procedure is derived from the standard one (`epx_evol_start`), but with the following modifications:

- No on-line files (executable, manuals) are updated
- All compilations are done with `/optimize:1` (i.e. optimized versions 0 and 3 are NOT updated automatically!)
- The QuickWin version is NOT built!
- Do not build or test the manuals (just update the correspondig `.ttx` source files)

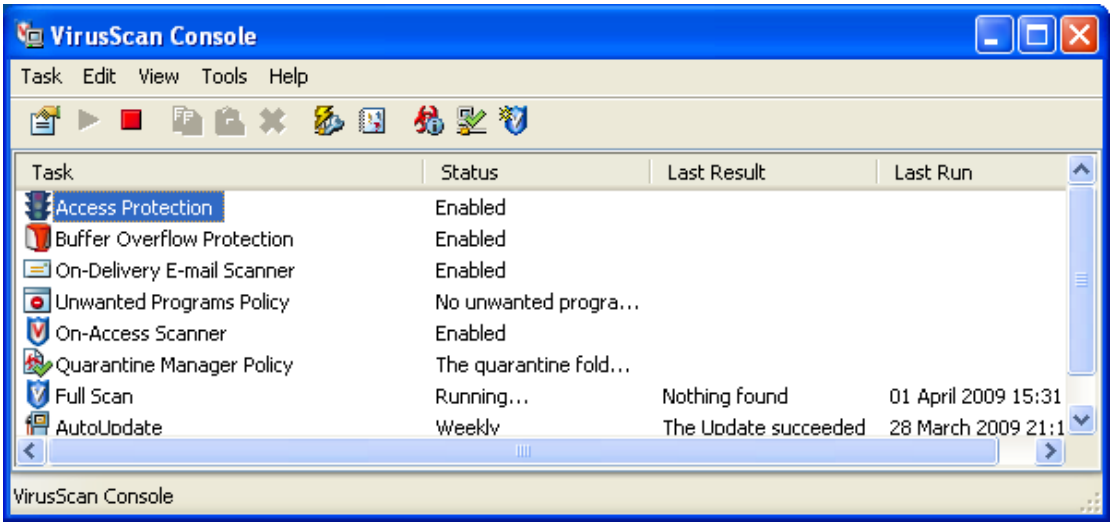
10.14.25 Solving a problem with automatic e-mail due to AntiVirus

In March 2009 the Antivirus software (McAfee) was updated on all machines, including the EUROPLEXUS server at JRC (\\sm47). The automatic mail message sent at the end of each evolution did not work any longer. Trying to send the message by epx_mail from a console did not work either: the error message (and also logged in file `epx_mail.err` in the EUROPLEXUS evolution directory) was:

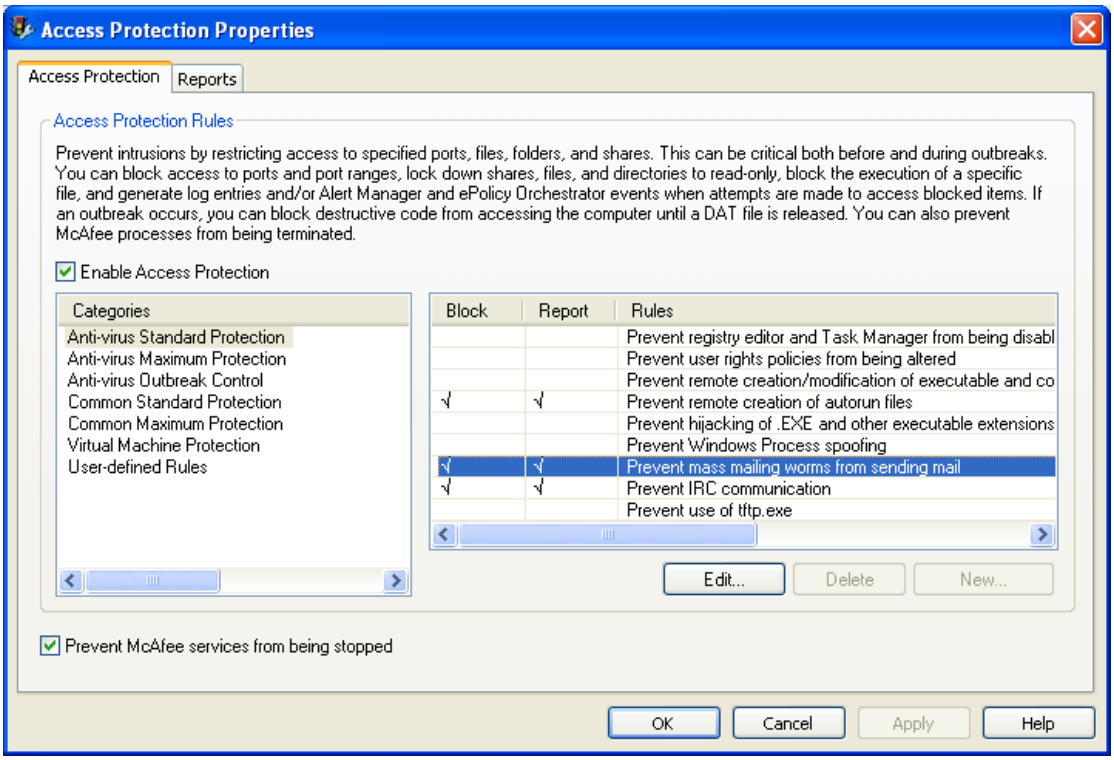
```
Error in connecting to isis-ms.jrc.it at port 25: Unknown error
```

After some inspection it was found that the problem is related to the new AntiVirus version (now 8.5). By default, potential spamming programs are prevented from sending mail, and Perl is considered to be one of these! To solve the problem, proceed as follows:

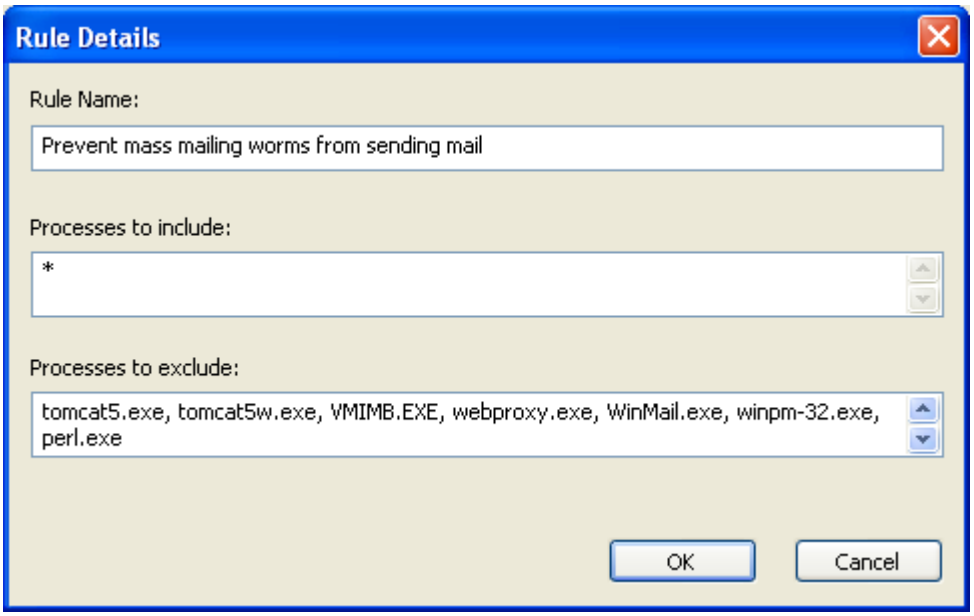
- Open the VirusScan console by right-clicking on the VirusScan icon and choosing the corresponding command:



- Click on Access Protection and select “prevent mass mailing worms from sending mail”:



- Click on Edit, then under "processes to exclude" add `perl.exe` separated by a comma plus a blank from the previous entry:



- Click on OK and quit

Now the `epx_mail` command should work.

At the same time, the epx_mail procedure has been updated, by replacing the obsolescent mail server name (isis-ms.jrc.it) by the new one (ipsc-mail.jrc.it) in the call to the SendMail package:

```
$sm = new SendMail("ipsc-mail.jrc.it");
```

Note also that with this correction, the automatic e-mail works also on the 64-bit machine, by using the same `epx_mail` procedure and the same `SendMail` package as on the 32-bit machines.

10.15 Installation under 64-bit operating system (Windows)

In September 2008 a new PC with 64-bit Windows operating system became available. The machine is a Siemens-Fujitsu PC with 4-CPU processor Intel Xeon 5110 @ 1.60 GHz, and 32 GB of RAM. The biggest advantage of this machine with respect to the 32-bit version is the large RAM memory, which should allow to perform much larger simulations than previously.

The operating system installed is MS-Windows XP Professional x64 Edition Version 2003, Service Pack 2. It is foreseen to subsequently install also a 64-bit Linux operating system in dual boot.

The compilation tools installed are:

- MS Visual Studio 2005 (version 8.0.50727.762) with MS .NET Framework Version 2.0.50727 SP1
- Intel Fortran compiler 64, Version 10.1.014, Build 20080112.

10.15.1 Porting to 64-bit

Compilation issues

In order to compile EUROPLEXUS under the 64-bit operating system, we start by specifying the `WIN` and `SPLIB` filtering keywords, but not the `OGL` keyword. In this way, a version without the OpenGL-based graphics is produced. This is because problems are expected with the graphical libraries under the 64-bit operating system (see below).

No special filtering keywords should be needed any more for the 64-bit version, since at CEA H. Bung compiles under Linux 64-bit without any special keyword.

To compile under 64-bit, the following modifications are performed:

- The `epx_setvars.bat` script is modified, by replacing the following line:

```
SET IFV=C:\Program Files\Intel\Compiler\Fortran\10.0.25\IA32\Bin\ifortvars.bat

SET IFV=C:\Program Files (x86)\Intel\Compiler\Fortran\10.1.014\em64t\Bin\ifortvars.bat
```

- The `epx_filter.f` and the `ordo.f` utility programs are re-compiled under the 64-bit operating system (thus obtaining `epx_filter.exe` and `ordo.exe`):

```
epx_setvars
ifort epx_filter.f
ifort ordo.f
```

- The `epx_cmp.pl` script is modified, by adding the following optimization options (the command without options corresponds to no optimization, i.e. the same as `-0`, while the command with the `-o` option corresponds to full optimization, i.e. the same as `-5`; `-1` and `-2` are identical and correspond to `/O1`, `-3` and `-4` are identical and correspond to `/O2`, `-5` corresponds to `/O3`):

```
-0 | -1 | -2 | -3 | -4 | -5
```

- After re-generating the `epx_cmp.bat` script by the command `pl2bat epx_cmp.pl`, do not forget to insert by hand the variable-setting lines at the beginning of the batch file (see Section 10.14.12):

```
@echo off
call "epx_setvars.bat"
```

- Consequently, the whole code is re-compiled by a command such as (where `<n>` represents a digit between 0 and 5):

```
epx_cmp -<n> -x -k WIN -k SPLIB
```

During the compilation, no errors arise.

Link issues

The epx_lk.pl script is modified by taking a default stack size of 1 GB (0x64000000) instead of 10 MB (0x640000). This is consistent with the fact that the 64-bit version of the code is mainly used for large (i.e. memory-intensive) applications.

Then, we need to produce new versions of the BLAS and SPLIB libraries (Libblas.lib and Libsp.lib, respectively):

- The BLAS library is obtained via the following make_all.pl script executed in the Blas_source directory (the only change with respect to the 32-bit version is that df -c is replaced by ifort /c):

```
system ("del *.obj");
system ("ifort /c /optimize:5 libblas.f");
system ("del ..\\library\\Libblas.lib");
system ("lib /OUT:Libblas.lib *.obj");
system ("del *.obj");
system ("move Libblas.lib ..\\library");
exit;
```

- The SPLIB library is obtained via the following make_all.pl script executed in the Splib_source directory (the only change with respect to the 32-bit version is that df -c is replaced by ifort /c):

```
system ("del *.obj");
system ("ifort /c /optimize:5 *.f");
system ("del ..\\library\\Libsp.lib");
system ("lib /OUT:Libsp.lib *.obj");
system ("del *.obj");
system ("move Libsp.lib ..\\library");
exit;
```

Note that both above libraries are compiled with the maximum possible optimization (/optimize:5).

In order to link the code (without the OpenGL-based graphics), the following modifications are performed:

- The epx_lk.pl script is modified, by replacing the following line (thus completely omitting the OpenGL-related libraries):

```
$libogl = "f90gl.lib f90glu.lib f90glut.lib glut32.lib bmplib.lib user32.lib gdi32.lib
vfw32.lib";

$libogl = "";
```

- After re-generating the epx_lk.bat script by the command pl2bat epx_lk.pl, do not forget to insert by hand the variable-setting lines at the beginning of the batch file (see Section 10.14.12):

```
@echo off
call "epx_setvars.bat"
```

- Finally, the whole code is linked by a command such as:

```
epx_lk -o
```

During the link, no errors arise.

Standard libraries

By searching on the C:\ directory in the 64-bit machine the libraries needed for the link, the following results are obtained:

- As concerns vfw*.lib, the following files are found:

```
C:\Program Files (x86)\Microsoft Visual Studio 8\VC\PlatformSDK\Lib\Vfw32.lib
C:\Program Files (x86)\Microsoft Visual Studio 8\VC\PlatformSDK\Lib\AMD64\Vfw32.lib
```

- As concerns gdi*.lib, the following files are found:

```
C:\Program Files (x86)\Microsoft Visual Studio 8\VC\PlatformSDK\Lib\Gdi32.lib
C:\Program Files (x86)\Microsoft Visual Studio 8\VC\PlatformSDK\Lib\GdiPlus.lib
C:\Program Files (x86)\Microsoft Visual Studio 8\VC\PlatformSDK\Lib\AMD64\Gdi32.lib
C:\Program Files (x86)\Microsoft Visual Studio 8\VC\PlatformSDK\Lib\AMD64\GdiPlus.lib
```

- As concerns `user*.lib`, the following files are found:

```
C:\Program Files (x86)\Microsoft Visual Studio 8\VC\PlatformSDK\Lib\User32.lib
C:\Program Files (x86)\Microsoft Visual Studio 8\VC\PlatformSDK\Lib\UserEnv.lib
C:\Program Files (x86)\Microsoft Visual Studio 8\VC\PlatformSDK\Lib\AMD64\User32.lib
C:\Program Files (x86)\Microsoft Visual Studio 8\VC\PlatformSDK\Lib\AMD64\UserEnv.lib
```

- As concerns `*gl*.lib`, the following files are found:

```
C:\Program Files (x86)\Microsoft Visual Studio 8\VC\PlatformSDK\Lib\GlAux.lib
C:\Program Files (x86)\Microsoft Visual Studio 8\VC\PlatformSDK\Lib\GlU32.lib
C:\Program Files (x86)\Microsoft Visual Studio 8\VC\PlatformSDK\Lib\OpenGL32.lib
C:\Program Files (x86)\Microsoft Visual Studio 8\VC\PlatformSDK\Lib\AMD64\GlAux.lib
C:\Program Files (x86)\Microsoft Visual Studio 8\VC\PlatformSDK\Lib\AMD64\GlU32.lib
C:\Program Files (x86)\Microsoft Visual Studio 8\VC\PlatformSDK\Lib\AMD64\OpenGL32.lib
```

- As concerns `*gl*.dll`, the following files are found:

```
C:\WINDOWS\System32\glu32.dll
C:\WINDOWS\SysWOW64\glu32.dll
C:\WINDOWS\ServicePackFiles\amd64\glu32.dll
C:\WINDOWS\System32\opengl32.dll
C:\WINDOWS\SysWOW64\opengl32.dll
C:\WINDOWS\ServicePackFiles\amd64\opengl32.dll
```

Execution issues

The first excecutable has been tentatively obtained (without the OpenGL graphics) by a fully-optimized compilation (`epx_cmp -5 …`). However, when the standard benchmarks are run with this version, a relatively large number (70 or so) tests fail. Most of these do not pass the qualification, but a few of them also produce a traceback. However, this might be due simply to the fact that the solution starts to diverge.

This fact is new because H. Bung reports that under Linux he had run all tests successfully with full optimization, but this was with version 9.1 of the Intel Fortran compiler. When using Version 10.1, he also gets similar optimization problems under Linux.

The alternative, at least for the moment (until perhaps the problem is solved by a later compiler version?) is to use a lower optimization, or no optimization at all (although this is not very satisfying):

- With `-0` (no optimization) all benchmarks pass;
- With `-1` or `-2` (same as `/O1`) just two benchmarks fail (`BM_IMP_PINB23` and `BM_IMP_PINL23`), with an error saying that the time step becomes too small: the test is then stopped prematurely and, consequently, the qualification is not passed.
- With `-3` or higher a large number of tests fail (about 70 or more).

Interestingly, by running the 32-bit executable (taken from the standard version on the 32-bit machine) on the 64-bit machine, all benchmarks pass (and this by including of course also the OpenGL-based graphical capabilities). However, in this case only the standard 32-bit memory can be addressed, and not the full 32 GB memory available on the 64-bit machine.

In Section 0 below, the possibility of de-bugging under the 64-bit operating system is explored.

GLUT library

The first graphical library to be ported to 64-bit is the GLUT library. We have tentatively used the same version in use for the 32-bit platform (based upon version 3.7.6 of 8 November 2001), because this version includes several modifications introduced for EUROPLEXUS (see Section 10.14.18). We have just tried to re-compile it under the 64-bit platform.

To this end, we have directly used the modified source files from directory `E:\EUROPLEXUS\Glut` on the 32-bit machine (these are the same as files in the package `Glut_f90gl_modifs.tar.gz`). The steps are as follows:

- On the 64-bit machine, create a directory, say `Glut`, under `E:\EUROPLEXUS`.

- Copy into this directory all the C source files from the (modified) Glut distribution, namely the files `glut-3.7.6\lib\glut\*.c`, with the following exceptions: files `capturexfont.c`, `glut_menu.c`, `glut_menu2.c` and `layerutil.c` (these files are unused).

- Copy into this directory all the C header files from the Glut distribution, namely the files `glut-3.7.6\lib\glut\*.h` (files `glutbitmap.h`, `glutint.h`, `glutstroke.h`, `glutwin32.h`, `layerutil.h`, `stroke.h`, `win32_glx.h` and `win32_x11.h`).

- Copy into this directory all the C header files `glut-3.7.6\include\GL\glut.h` and `glut-3.7.6\include\GL\glutf90.h`.

- Copy the two just mentioned header files `glut.h` and `glutf90.h` also to the following directory (this directory should already contain the other basic OpenGL include files `GL.h`, `GLAux.h` and `GLU.h`), otherwise the header files are not found during the following C compilation (see next step):

  `C:\Program Files (x86)\Microsoft Visual Studio 8\Vc\PlatformSDK\Include\gl`

- Compile all files and produce the library `Glut32.lib`, e.g. by first invoking the `epx_setvars` script (to set the environment), and then by using the following perl script (`make_all.pl`):

```
system ("del *.obj");
system ("del Glut32.lib");
system ("cl /c /o2 *.c");
system ("lib /OUT:Glut32.lib *.obj");
system ("del *.obj");
exit;
```

- Install the library by manually copying the file `Glut32.lib` to the following directory, where also the other 64-bit libraries are believed to reside (see "Standard libraries" above):

  `C:\Program Files (x86)\Microsoft Visual Studio 8\Vc\PlatformSDK\Lib\AMD64`

The above compilation is believed to build up a 64-bit library, but this has to be further checked!

F90gl library

The second graphical library to be ported to 64-bit is the F90GL library. We have tentatively used the same version in use for the 32-bit platform (based upon version 1.2.8 of F90GL), because this version includes several modifications introduced for EUROPLEXUS (see Section 10.14.18). We have just tried to re-compile it under the 64-bit platform.

To this end, we have directly used the modified source files from directory `E:\EUROPLEXUS\F90gl` on the 32-bit machine (these are the same as files in the package `Glut_f90gl_modifs.tar.gz`). The steps are as follows:

- On the 64-bit machine, create a directory, say `F90gl`, under `E:\EUROPLEXUS`.

- Copy into this directory all the files (and sub-directories) from the corresponding directory on the 32-bit machine.

The main directory contains makefiles for the various supported platforms. The one for MS Windows and the Intel compiler is the batch file `mf8nio.bat` Note that the files with an extension `.fpp` are Fortran 90 files that need to be processed by a pre-processor (`sppr.exe`) that comes with the standard `f90gl` distribution (see under `f90gl\util`).

This script (in directory `f90gl-1.2.8`) has to be modified as follows:

- change the value of the WININC variable from: set WININC=F:\Program Files\Microsoft SDK\include to: set WININC=c:\Program Files (x86)\Microsoft Visual Studio 8\Vc\PlatformSDK\Include.

Then in order to compile and build up the standard version, first invoke the epx_setvars command to set the environment and then just type:

```
mf8nio
```

This will produce the following library files:

- f90gl-1.2.8\lib\f90GL.lib
- f90gl-1.2.8\lib\f90GLU.lib
- f90gl-1.2.8\lib\f90GLUT.lib

Copy these files to the Intel compiler's em64t\LIB directory (on the author's machine, this is C:\Program Files (x86)\ Intel\Compiler\Fortran\10.1.014\em64t\Lib).

The following eight Fortran module files are also produced:

- f90gl-1.2.8\lib\opengl_fwrap.mod
- f90gl-1.2.8\lib\opengl_gl.mod
- f90gl-1.2.8\lib\opengl_glinterfaces.mod
- f90gl-1.2.8\lib\opengl_glu.mod
- f90gl-1.2.8\lib\opengl_gluinterfaces.mod
- f90gl-1.2.8\lib\opengl_glut.mod
- f90gl-1.2.8\lib\opengl_glutinterfaces.mod
- f90gl-1.2.8\lib\opengl_kinds.mod

Copy these files to the Intel compiler's em64t\Include directory (on the author's machine, this is C:\Program Files (x86)\Intel\Compiler\Fortran\10.1.014\em64t\Include).

If during the execution of the mf8nio script you get compilation errors (e.g. in cwrapglt.c), make sure you have copied the glutf90.h include file as explained in the previous Section.

Bmplib library

The library was re-built for 64-bit by copying the previous (32-bit) solution to a directory E:\EUROPLEXUS\Bmplib, opening the solution (which converts it automatically) and building it.

However, note that in this way the solution is built for 32-bit, not for 64-bit. Use the Configuration Manager to edit the existing Debug and Release configurations of the Visual Studio solution by activating the x64 platform in place of the Win32 platform. This is rather tricky and not fully understood, see also "SET THE X64 PLATFORM" in Section 10.15.2.

At the end, the library file bmplib.lib (Release version) is copied by hand to the directory C:\Program Files (x86)\Intel\Compiler\Fortran\10.1.014\em64t\Lib).

Another alternative to produce the bmplib library is to use a script similar to the ones used for the other two libraries. First invoke the epx_setvars command to set the environment (this sets for 64-bit compilation) and then and then use the following perl script (make_all.pl):

```
system ("del *.obj");
system ("del Bmplib.lib");
system ("cl /c /o2 *.c");
system ("lib /OUT:Bmplib.lib *.obj");
system ("del *.obj");
exit;
```

Linking the version with OpenGL

Once all the above mentioned OpenGL-related libraries (`Glut32.lib`, `f90GL.lib`, `f90GLU.lib`, `f90GLUT.lib` and `Bmplib.lib`) have been produced in 64-bit version and placed properly in the system, we have compiled the EUROPLEXUS sources with the `OGL` keyword set (but still without optimization):

```
epx_cmp -0
```

and created the `Libplex.lib` library:

```
LIB /OUT:Libplex.lib *.obj
```

Finally, the code is linked by the modified `epx_lk` command (the one using a bigger stack size), i.e. with all the usual libraries:

```
$libogl = "f90gl.lib f90glu.lib f90glut.lib glut32.lib bmplib.lib user32.lib gdi32.lib
vfw32.lib";
```

At this point, link errors are produced because the linker seems unable to find the following symbols:

```
_PrepareTheBitmap
_FinishTheBitmap
_ReadTheBitmap
_LoadTheBitmap
_PrepareTheAVI
_WriteAviFrame
_FinishTheAVI
```

All these functions are contained in the Bmplib library, and are called from the Fortran90 `M_RENDER.ff` module via `INTERFACE` declarations. These are of the form (for example):

```
INTERFACE
SUBROUTINE PrepareTheBitmap (w, h)
!DEC$ ATTRIBUTES C, ALIAS:'_PrepareTheBitmap' :: PrepareTheBitmap
INTEGER :: w, h
END SUBROUTINE PrepareTheBitmap
END INTERFACE
```

The problem seems related to the fact that the ALIAS declaration contains an underscore prepended to the subroutine name, which is not recognized.

By removing this undersore from all `ALIAS` declarations in the `M_RENDER.ff` module and re-compiling `M_RENDER.ff`, the link is executed without errors.

It has been verified that, when producing the 32-bit version of EUROPLEXUS, the underscores are needed. In fact, the compiler documentation specifies that the convensions are diffierent between the two platforms!

The resulting 64-bit executable has been tested and all graphical functionalities (interactive screen rendering, bmp files, animations) seem to work correctly (at least in this version compiled without any optimizations).

In order to obtain a unique source for both platforms of `M_RENDER.ff`, the only way seems to be using the filtering process (`epx_filter` programme, see Section 5.1). However, some tests have

revealed bugs in the current version of the epx_filter programme: in particular, only one CELIF branch per conditional may be used, contrary to what says the documentation.

Therefore:

- The epx_filter programme is corrected by eliminating the above bug (now more than one CELIF branch is accepted) and another minor bug (wrong exit code in a specific error case).
- Two new filtering keys are introduced: W32 for the Windows 32-bit platform and W64 for the Windows 64-bit platform. These are at the moment used only in module M_RENDER.ff. Note that the WIN keyword must also be always specified, for Windows platforms (32 or 64 bit alike).
- The relevant scripts (epx_cmp, epx_filbat etc.) are updated accordingly, on both platforms.

Therefore the standard filtering keys for EUROPLEXUS become as follows:

- For Windows 32-bit : WIN OGL W32 SPLIB;
- For Windows 64-bit : WIN OGL W64 SPLIB.

The corrected module M_RENDER.ff will be evolved only after the filtering programme has been updated on all mirror sites.

Executing the version with OpenGL

Here are some statistics about the execution of the full EUROPLEXUS version (i.e. with OpenGL graphics) under the 64-bit platform, depending upon optimization. All values refer to code version #1564 of 7 October 2008, containing 638 non-regression benchmark tests.

- Without optimization, all benchmarks pass by using 61 minutes of CPU. For comparison, the same benchmarks on the 32-bit platform acting as a mirror site takes 57 minutes (with /optimize:5), but that computer is less powerful (Pentium 4 with single CPU at 2.0 GHz and 1 GB of RAM). Finally, the benchmarks on the 32-bit platform SM61 (Pentium 4 with 2 CPUs at 3.0 GHz and 1 GB of RAM) takes 46 minutes.
- With /optimize:1 the following two benchmarks fail: BM_IMP_PINB23 and BM_IMP_PINL23. The execution takes 34 minutes of CPU. However, further 120 benchmarks show small differences (but still within validation tolerances) in the PostScript plots, when compared to the results obtained on the same platform without optimization.
- With /optimize:3 the execution takes 30 minutes of CPU. A total of 64 benchmarks fail for various reasons: mostly due to qualification errors, but 5 of these generate and "access violation" and a traceback (but this may be due to strong divergence of the solution towards unphysical results). The latter are: BM_DOM_2D_INCMOD, BM_DOM_3D_SHELL_INCMOD, BM_DOM_MODA_UTIL, BM_DOM_MODA_SAM and BM_STR_SNC_FRAG. A total of 545 benchmarks show differences in the PostScript plots, when compared to the results obtained on the same platform without optimization.

10.15.2 Preparing a solution under Visual Studio 2005 (64-bit version)

In order to develop EUROPLEXUS under Microsoft Visual Studio 2005 (64-bit version), set up a new Solution (say plex) containing two Projects: filter and epx. The filter project contains the unfiltered "*.ff" Fortran source files, while the epx project contains the corresponding filtered sources "*.f", which may actually be compiled. The filtering process is automatically performed since it is set up as a "custom build step" action. Hereafter, we describe the set up of a Solution of type Console application, which is the standard for EUROPLEXUS (since May 2003).

To set up the Solution, do the following:

1. **CREATE AN EMPTY SOLUTION** - Open Microsoft Visual Studio 2005 (***Start → All Programs → Microsoft Visual Studio 2005 → Microsoft Visual Studio 2005***). This opens up the "Microsoft Visual Studio – Start Page" window. Click on menu ***File → New → Project***. This opens up the New Project dialog box. Under Project types select ***Other Project Types →Visual Studio Solutions → Blank Solution***.

   Under ***Name*** type in `plex`, and under ***Location*** type your directory of choice (e.g.: `E:\Europlexus\Test_debug`) and then click on ***OK***. This creates a (new) sub-directory `E:\ Europlexus\Test_debug\plex` containing the solution file `plex.sln` and an associated file `plex.suo` (the latter appearing as grayed out).

   Click on ***File → Save All*** to save the work done so far (do this from time to time).

2. **ADD A "FILTER" PROJECT TO THE SOLUTION** - A "Solution Explorer" pane appears in the right upper part of the application window, that contains "Solution 'plex' (0 projects)". Right-click on this sentence to show a pop-up menu and click on ***Add → new project***.

   The "Add New Project" dialog appears, under Project Types choose ***Visual C++ → Empty Project***, in the ***Name*** box type `filter` and in the ***Location*** box leave `E:\Europlexus\Test_debug\plex` which should appear already as such.

   By clicking on ***OK***, the new project `filter` should appear in the Solution Explorer under "Solution 'plex' (1 project)", with three sub-entries (empty for the moment): Header Files, Resource Files and Source Files. A new file `plex.ncb` and a new sub-directory `filter` are created under `E:\Europlexus\Test_debug\plex`. The latter should contain a file called `filter.vcproj`.

   Right-click on the `filter` project in the Solution Explorer to show the pop-up menu and choose ***Properties***. This opens up the "filter Property Pages" dialog. In the upper part of the dialog, under ***Configuration***, make sure that you choose ***All Configurations***, so that the subsequent settings will apply to both standard configurations of the project (Debug and Release): although only the Debug configuration will actually be used for this project, it is better to avoid confusion.

   Under ***Configuration Properties → General → Project Defaults → Configuration type*** appears "Application (exe)". This entry should be changet into "Utility": to do so, click on it until a drop-down menu appears (a down arrow) and then choose ***Utility*** from the menu. Then click on the ***Apply*** button. In this way all the entries (Linker, Web deployment, etc.) that were previously present under Configuration Properties should disappear and the only remaining ones should be: General, Debugging and Build Events. Click on ***OK*** to close the dialog.

   ADD FILES TO THE FILTER PROJECT - Now let's us add an (unfiltered) source file to the `filter` project. Open a Console window on directory `E:\Europlexus\Test_debug \plex` and type command `epx_get main`: this will copy the EUROPLEXUS main program `main.ff` from the sources library to this directory. Next, in the Solution Explorer, right-click on ***filter → Source Files*** to make the pop-up menu appear, and choose ***Add → Add Existing Item***. The "Add existing Item – filter" dialog appears, already set with ***Look in: filter***. Change it to the parent directory by clicking on the appropriate icon (a folder icon with an upward arrow) so that it becomes ***plex***, and under ***Files of type*** choose ***All Files (*.*)***. Alternatively, under ***File name*** you may enter ****.ff***, so that only the EUROPLEXUS source files are visualized. Select `main.ff` by clicking on it, then click the ***Open*** button. The file `main.ff` should now appear under ***filter → Source Files*** in the Solution Explorer. A dialog box "Matching Custom Build Rule Not Found" pops up, asking if you want to create a new rule file to define a custom build rule to build files with this extension. You have two alternatives:

   - Click on ***No***. In this way, however, you will have to set manually the properties for each new file `.ff` that you add to the project (see Section 10.14.16). Follow the instructions under "MANUALLY SET THE FILTERING PROPERTIES" below.

- Click on ***Yes***. In this way a dialog appears that allows you to create a new rule for files with a `.ff` extension. Follow the instructions in Section 10.14.16. The advantage is that, whenever you add a new `.ff` file to the `filter` project, the settings are done automatically according to the rule instead of having to set them manually. Skip the following paragraph "SET THE FILTERING PROPERTIES" (but make sure you have a file `epx_filbat.bat` as explained therein).

MANUALLY SET THE FILTERING PROPERTIES - Next, in the case one has chosen not to define any rule, we must set the filtering for the newly added file. Right-click on the file name (`main.ff`) in the Solution Explorer to open the pop-up menu, and choose ***Properties***. This opens the "main.ff Property Pages" dialog. Under ***Configuration***, make sure you choose ***All Configurations***, then click under ***Configuration Properties → Custom Build Step*** and under ***Command Line*** insert the following string:

```
%UTILDIR%\epx_filbat.bat "$(InputDir)""$(InputName)"
```

Note that there is a space between '`bat`' and the first double quote but there should be no space between the two contiguous double quotes in the above command! For the meaning of the `UTILDIR` environment variable, see Section 10.13.23.

The `epx_filbat.bat` file contains the following simple commands:

```
@echo off
echo Filtering %1.ff
%UTILDIR%\epx_filter.exe WIN OGL SPLIB %2 %3 <%1.ff >%1.f
echo Filtering done
```

Under ***Outputs***, insert:

```
$(InputDir)$(InputName).f
```

Then click on the ***Apply*** button and finally on the ***OK*** button to close the dialog.

This has the effect of invoking the `epx_filter` utility described in Section 5.1 with the three (hard-coded) passwords `WIN OGL SPLIB` on file `$(InputDir)\$(InputName).ff` and placing the filtered output in file `$(InputDir)\$(InputName).f`. Here `$(InputDir)` is `E:\Europlexus\Test_debug\plex` and `$(InputName)` is for example `main`.

Having selected multiple Source Files entries (by shift-clicking them) rather than a single file (`main.ff`) when applying the Custom Build setting, has the effect that the custom build action is associated to all files currently selected. This is important to speed up the operation when many files are present in the project. However, unfortunately the operation has to be repeated anew whenever new files are added to the project.

TEST OUT THE FILTERING PROCESS - To test out the correct functioning of the `filter` project, right-click on the `main.ff` file under Solution Expolorer, and from the pop-up menu choose ***Compile***. This should trigger the filtering of the file to produce `main.f`, that will be located in the same directory as `main.ff`, i.e. under `E:\Europlexus\Test_debug\plex`. A log of the filtering process should appear in the ***Output*** pane of the application window and also as a file named `Buildlog.htm` in the `filter\Debug` sub-directory. The messages appearing in the Output pane should be similar to the following:

```
------ Build started: Project: filter, Configuration: Debug x64 ------

Performing Custom Build Step
Filtering "e:\Europlexus\Test_debug\plex\""main".ff
1
Filtering done

Build log was saved at "file://e:\Europlexus\Test_debug\plex\filter\Debug\
BuildLog.htm"
filter - 0 error(s), 0 warning(s)
```

```
---------------------- Done ----------------------

    Build: 1 succeeded, 0 failed, 0 skipped
```

3. **ADD A COMPILATION PROJECT (EPX) TO THE SOLUTION** - It is now time to add the second project to the solution, i.e. the `epx` project, that will perform the actual compilation and link to produce the EUROPLEXUS executable.

   In the Solution Explorer, right-click on the "Solution 'plex' (1 project)" to show the pop-up menu and click on ***Add → new project***. The "Add New Project" dialog appears, under Project Types choose ***Intel® Fortran → Console Applications → Empty project***, in the ***Name*** box type `epx` and in the ***Location*** box leave `E:\Europlexus\Test_debug\plex` which should appear already as such.

   A new project `epx` should appear in the Solution Explorer pane, which should change to "Solution 'plex' (2 projects)". The new project contains three (empty) sub-entries: Header Files, Resource Files and Source Files. A new sub-directory `epx` is created under `E:\Europlexus\Test_debug\plex`. This should contain a file called `epx.vfproj`.

   ADD FILES TO THE EPX PROJECT - Now let's us add the previously obtained (filtered) source file `main.f` to the `epx` project. In the Solution Explorer, right-click on ***epx → Source Files*** to make the pop-up menu appear, and choose ***Add → Add Existing Item***. The "Add existing Item – filter" dialog appears, already set with ***Look in: epx***. Change it to the parent directory by clicking on the appropriate icon (a folder icon with an upward arrow) so that it becomes ***plex***, and make sure that under ***Files of type*** the ***Common Intel® Fortran Files*** type is selected (it should be automatically so). Alternatively, under ***File name*** you may enter ****.f***, so that only the EUROPLEXUS filtered source files are visualized. Select `main.f` by clicking on it, then click the ***Add*** button. The file `main.f` should now appear under ***epx → Source Files*** in the Solution Explorer.

   SET PROJECT DEPENDENCIES - Finally, we should specify that the `epx` project "depends" upon the `filter` project. Select the menu ***Project → Project Dependencies*** and under the Dependencies tab in the Projects selector select `epx`, then in the Depends on selector check out the box corresponding to the `filter` project. Conversely, the `filter` project should not depend upon `epx` (otherwise a circular dependency would be created), so the corresponding check box should appear grayed out. In the Build Order pane, make sure that "Projects build in this order" contains first `filter` and then `epx`. Then click on the ***OK*** button to close the dialog.

4. **ADJUST THE SETTINGS FOR THE EPX PROJECT** - Right-click on the `epx` project in the Solution Explorer to show the pop-up menu and choose ***Properties***. This opens up the "epx Property Pages" dialog. In the upper part of the dialog, under ***Configuration***, make sure that you choose ***All Configurations***, so that the subsequent settings will apply to both standard configurations of the project (Debug and Release): although the Debug configuration will most often be used for this project, it is better to avoid confusion.

   Under ***Configuration Properties → Fortran → General*** set Additional Include Directories to the following:

   ```
   $(EUROPLEXUS)\Include;$(EUROPLEXUS)\Module
   ```

   and then click on ***Apply***. This has as an effect that include files (`*.inc`) and object module files (`*.mod`) which are not found on the current directory are searched in the two listed directories before the default directories.

   Under ***Configuration Properties → Fortran → Data*** set "Local Variable Storage" to ***Local Variables Automatic*** instead of ***Default Local Storage***. Click on ***Apply***.

   Under ***Configuration Properties → Fortran → Run-time*** set "Generate Traceback Information" to ***Yes*** instead of the default (which appears to be Yes for the Debug configuration and No for the Release configuration). Click on ***Apply***. This generates an interactive warning message which may

be ignored. However, the following warning message appears: Microsoft Development environment – Traceback functionality does not work when the Omit Frame Pointers optimization is set to Yes in the Fortran Optimization category or when Enable Incremental Linking is set to Default or Yes in the Linker General category.

Under ***Configuration Properties → Fortran → Run-time*** set "Check Array and String Bounds" to ***No*** instead of the default (which appears to be Yes for the Debug configuration and No for the Release configuration). This is necessary due to the variable passing strategy of EUROPLEXUS coming from its ancestors). Click on ***Apply***.

Debug information for Fortran PARAMETER constants is activated by clicking on the `epx` project to make sure it is selected, then selecting ***Configuration Properties → Fortran → Debugging*** and by setting ***Information for PARAMETER Constants*** to ***All*** instead of None (which is the default). This corresponds to command-line compiler switch `/debug-parameters:all`. Note, however, that this setting must be done for the Debug configuration only (i.e., not for All Configurations!)

NOTE: the following stack initialization setting is NOT done if the OpenGL graphics is included in the project, see Section 10.14.19, otherwise the same problem noted in that Section is encountered. Trapping of unitialized variables is activated by clicking on the `epx` project to make sure it is selected, then selecting ***Configuration Properties → Fortran → Data*** and by setting ***Initialize stack values to an unusual value*** to ***Yes*** instead of No (which is the default). This corresponds to command-line compiler switch `/Qtrapuv` (and must be done for All Configurations).

Under ***Configuration Properties → Linker → General*** set "Additional Library Directories" to the following:

```
$(EUROPLEXUS)\Library
```

and then click on ***Apply.***

Under ***Configuration Properties → Linker → Input*** set "Additional Dependencies" to the following:

```
Libplex.lib Libblas.lib Libsp.lib f90gl.lib f90glu.lib
f90glut.lib bmplib.lib user32.lib gdi32.lib vfw32.lib
```

and then click on ***Apply.***

Under ***Configuration Properties → Linker → System*** set "Stack Reserve Size" to ***65536000*** (this corresponds to 1 GB, as mentioned above when dealing with modifications in the `epx_lk.pl` script) instead of the default value of 0 and then click on ***Apply***.

5. **DEFINE THE STARTUP PROJECT** - Next, we must set the `epx` project as the startup project for the solutioin, so that it is executed when debugging is launched. Right-click on "Solution 'plex' (2 projects" and select ***Set Startup Projects***. The "Solution 'plex' Property Pages" dialog opens up.

   In the ***Common Properties → Startup Project*** pane set the Single Startup Project radio button and select `epx` (not `filter`) as the startup project, click on ***Apply*** and then on ***OK***.

   SET THE X64 PLATFORM – When you build a new project on the 64-bit machine, by default the chosen platform is the Win32 platform, so if you try compiling and linking your Solution you will get errors because the compiler does not recognize the compiled module files (`.mod`) and libraries (`.lib`), saying that they have been built by an invalid version of the compiler!

   To solve this problem, the x64 platform must be set for the projects in the Solution. Right-click on the `plex` solution in the Solution Explorer to show the pop-up menu and choose ***Properties***. This opens up the "Solution 'plex' Property Pages" dialog. In the upper part of the dialog, under

Configuration, make sure that you choose ***All configurations***, so that the subsequent settings will apply to all configurations.

In the left pane click on ***Configuration Properties → Configuration***. Two projects appear in the table on the right (`epx` and `filter`). You must make sure that in the Platform column, x64 (and not Win32) is listed, for both projects. This may be somewhat tricky: if this platform is not listed anywhere in the drop-down menus, try using the Configuration manager button on the right to define it anew. Make sure that the Build checkbox is checked for both pojects, and at the end leave the x64 platform selected in the Platform list box at the top centre of tye dialog.

6. **TEST OUT THE SOLUTION** - The setting of the Solution is now complete. To test it, first remove the filtered file `main.f` from the `E:\Europlexus\Test_debug\plex` directory (to force its re-generation), then click on "Solution plex (2 projects)" to highlight the whole solution, make sure the current configuration is Debug, and finally choose ***Build → Rebuild Solution***. This should rebuild the entire solution, i.e. the two projects `filter` (filtering of `main.ff`) and `epx` (compilation of `main.f` and link). The messages should be something like:

```
1>------ Build started: Project: filter, Configuration: Debug x64 ------
1>Performing Custom Build Step
1>Filtering "e:\EUROPLEXUS\Test_debug\plex\""main".ff
1>1
1>Filtering done
1>Build log was saved at "file://e:\EUROPLEXUS\Test_debug\plex\filter\x64\Debug\
BuildLog.htm"
1>filter - 0 error(s), 0 warning(s)
2>------ Build started: Project: epx, Configuration: Debug x64 ------
2>Compiling with Intel(R) Fortran Compiler 10.1.014 [Intel(R) 64]...
2>main.f
2>Linking...
2>LINK : warning LNK4098: defaultlib 'LIBCMT' conflicts with use of other libs;
use /NODEFAULTLIB:library
2>Embedding manifest...
2>
2>Build log written to "file://E:\EUROPLEXUS\Test_debug\plex\epx\x64\Debug\
BuildLog.htm"
2>epx - 0 error(s), 1 warning(s)
========== Build: 2 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

A new executable `epx.exe` should be produced in directory `epx\x64\Debug`. This directory will also contain the object file `main.obj`, and three other auxiliary files (`epx.pdb`, `Buildlog.htm` and `epx.exe.intermediate.manifest`). Another log file `Buildlog.htm` appears in directory `filter\x64\Debug`.

Repeat the above build process also for the Release configuration. This should also work smoothly.

The setup of the Solution is now complete. Make sure you save your work by clicking on ***File → Save All*** before quitting the application.

10.15.3 Preparing the MPI (parallel) version of the code

Following some experimentation, from July 2009 the MPI version of the code (for Windows) is automatically built up at every evolution. To produce the MPI version, first the MPI package must be installed (on each machine that should be able to build and/or run this version).

To install MPI, go to the following internet site (to obtain the address, search for MPICH2): http://www.mcs.anl.gov/research/projects/mpich2/. Then download the Windows version: for 32-bit this is currently mpich2-1.1-win-ia32.msi, while for 64-bit it is mpich2-1.1-win-x86-64.msi.

Click on the executable and follow the instructions. A password (behappy) is proposed for access to execution, just accept it. Then make sure to check the "Install for everyone" box in the window "Select

Installation folder". The program is installed by default on folder `C:\Program Files\MPICH2`. If you get the following dialog:

then click on Unblock.

MPI Include (and Module) files

A sub-directory `include` is automatically created under the MPICH2 installation directory. An include directory `include_mpi` should then be created under the `$EUROPLEXUS` directory (normally on the EUROPLEXUS server, where also the normal `include` directory is located) and then all files (`*.h` and `*.mod`) from the `MPICH2\include` directory should be copied therein, so they are available for compilation from every machine However, it seems that as far as concerns the `.h` files, only `mpif.h` is actually used (this file contains FORTRAN statements, despite its extension).

EUROPLEXUS Module files

Furthermore, a `module_mpi` directory should also be created under the `$EUROPLEXUS` directory (normally on the EUROPLEXUS server, where also the normal `module` directory is located). This directory will hold all the `.mod` module files compiled by the `MPI` keyword.

MPI Library files

A sub-directory `lib` is automatically created under the MPICH2 installation directory. This contains, among others, two library files (`fmpich2.lib` and `mpi.lib`) which are needed during the linkage, and which therefore must be copied under the $EUROPLEXUS\library directory of the EUROPLEXUS server, so they are available for linking from every machine.

Generating the MPI executable

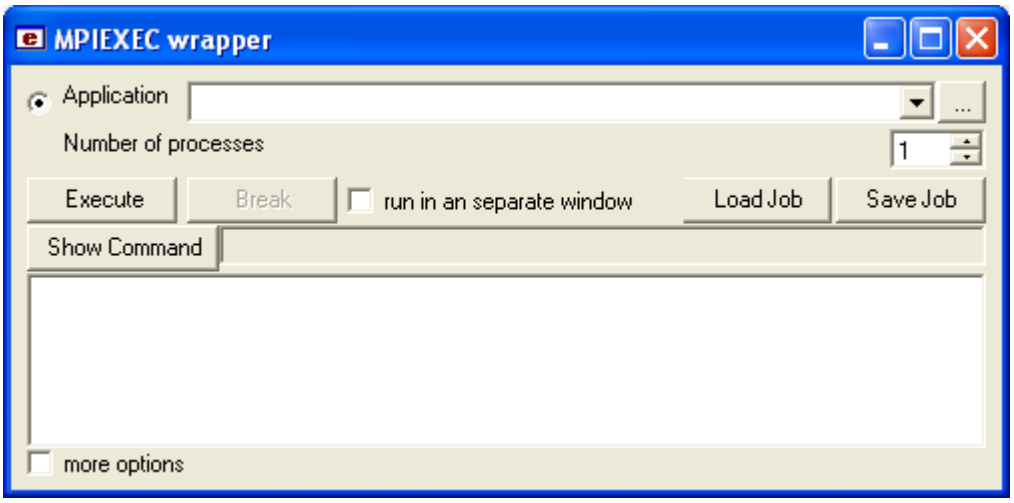
Then, to produce an MPI executable version of the code, use the following compilation and linking commands:

- `epx_cmp -o -M`
- `epx_lk -o -M`

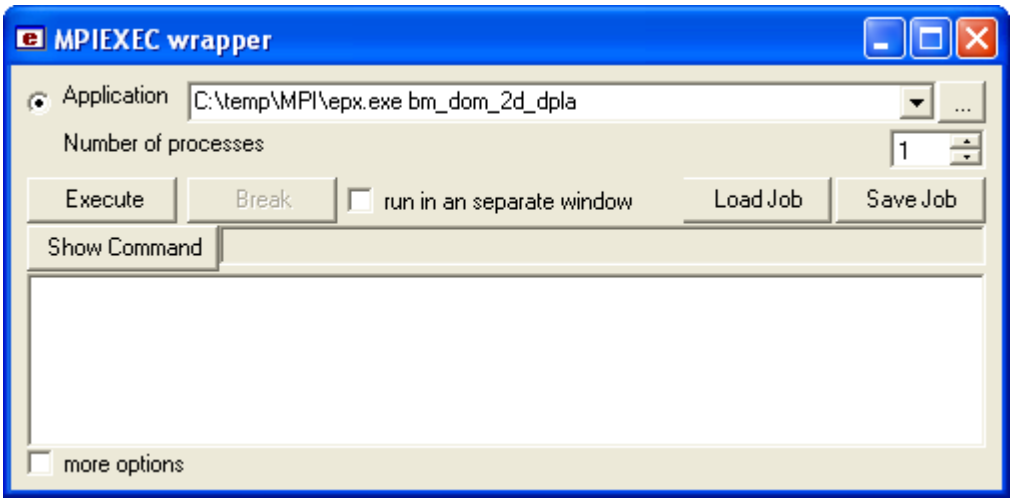
Running the MPI executable

To run the MPI executable from the current directory (say it is called `epx.exe`):

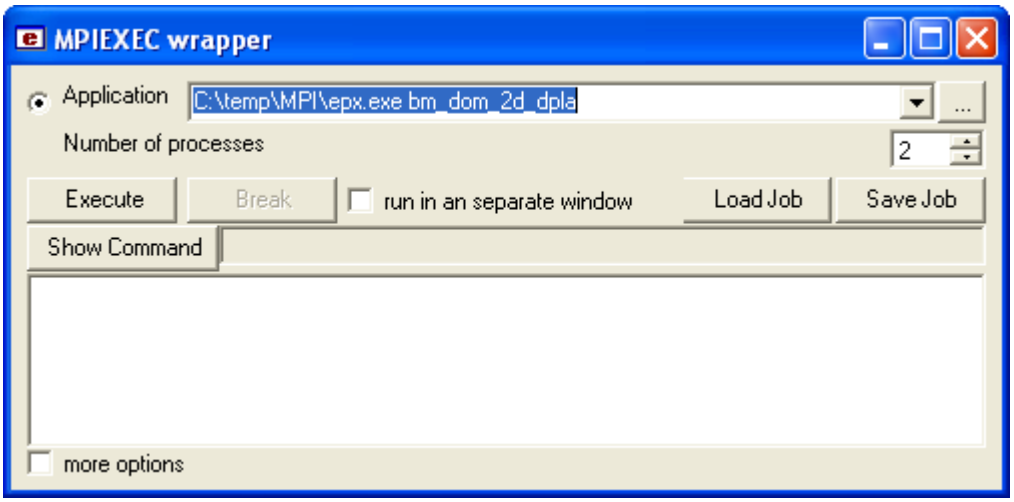
- Launch MPI by clicking on Start → All Programs → MPICH2 → wmpiexec.exe. A dialog window appears:



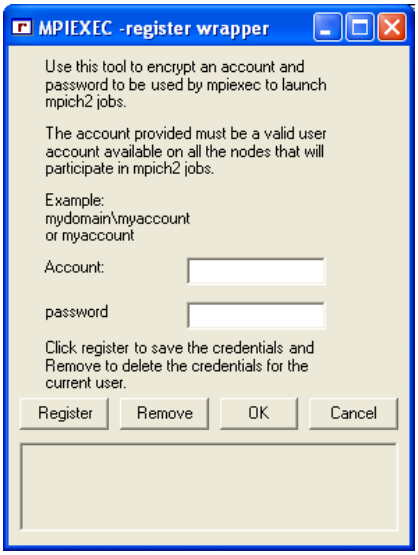
- Under Application, put the full path of the executable you want to run (e.g. navigate to it by clicking on the three dots) and then, separated by a blank, the name of the EUROPLEXUS input file to be run (this file must of course reside in the same subdirectory as the executable):



- Set the number of processors you want to use (e.g. 2). Of course, this must match exactly match the number of sub-domains defined in your EUROPLEXUS input file (in this example, we should have `STRUCTURE 2 ...`):



- Click on the Execute button
- The first time you use it, you may get the following dialog window (which can be also accessed by clicking on Start → All Programs → MPICH2 → wmpiregister.exe):



- Here insert your normal account (user name) and password, i.e. the values you normally use for logging in on your current workstation, and then press Register and finally OK.
- Once the execution starts, you should see appear in the text frame the output lines that normally come to the console.

10.15.4 Runtime check (-check) version of the code executable

From end July 2009 a fully-checked at runtime version of the code executable is built up at every evolution on the 32-bit platform. The executable is called `europlexus_check.exe`. To invoke this version, however, it is only necessary to use the `-c` switch in the `epx_bench` command:

```
epx_bench -c -l toto
```

To build up and use a local fully-checked at runtime version, proceed as follows:

- Make sure you have the `main.ff` file locally (if not, just retrieve it by `epx_get main`).

- Compile all local sources with the `-c` switch: this adds the `/check` option to the compilation switches: `epx_cmp -o -c`.
- Link with the `-c` switch: this links against the `-check` version of the EUROPLEXUS library (`libplex_check.lib`): `epx_lk -o -c`.
- Use the local `epx.exe` produced executable normally: `epx_test_benchmarks -l` or `epx_bench -e epx.exe -l toto`. Attention, you do *not* need the `-c` switch here (else it will use the standard `-check` executable and not the local executable).

Floating point checks

Following the implementation of the `-check` version at CEA, if the name of the file to be compiled is `main` the compilation procedure `epx_cmp` automatically adds the `/fpe:0` and `/fp:strict` switches which activate a stricter check of the floating point results. However, this generates a lot of errors in the OpenGL part of the code (i.e. in the libraries). The problem is that the floating point verification is a global setting (decided by the main program) and may not be set individually for each routine or library.

As a workaround to this problem, the `epx_cmp` command is modified in such a way that the `-c` switch only adds the `/check` (no floating point verification). A new switch `-C` is added, which allows to force the floating point verification in addition to the `/check`.

Automatic testing of all benchmarks by the –check option

A new procedure `epx_evol_check` has been prepared which executes the full benchmarks suite with the `-check` version. This command has been added in the scheduled tasks (and the `epx_schtasks` procedure has been modified accordingly) so that the tests are executed automatically every day.

The benchmarks are executed in a special directory `%EUROPLEXUS%\Check` and the errors of the last execution may be checked in a file named `_errors.log` in this directory. A trace of the launchings can be found in `%EUROPLEXUS%\Fromcentral\evol_check.log`.

Avoiding lengthy warning messages with the –check option

The `/chech:all` compilation option used to build the fully checked version of the code produces a huge number of warning messages concerning the creation of array temporaries in exchange lists. For example:

```
CALL toto (TRANSPOSE(m))
```

gives a warning. This can be avoided by using the following programming style:

```
tm = TRANSPOSE(m)

CALL toto (tm)
```

However, the first form is more compact and perhaps more redable. In order to be able to use this type of programming style, the checking of whether actual arguments are copied into temporary storage before routine calls is disabled by adding an extra switch to the compilation command:

```
/chech:all /check:noarg_temp_created
```

10.15.5 New optional switches for the epx_evol_start procedure

Three new optional switches have been added to the `epx_evol_start procedure`, which allows to bypass the build up of the QuickWin version (`-noqw`), of the MPI version (`-nompi`) and of the –check version (`-nocheck`). This considerably speeds up the evolution of the code, but should only be

used in emergency conditions, e.g. after a blockage of the local evolutions which has caused a long list of evolutions to be done. The complete syntax is then:

```
epx_evol_start [-p] [-noqw] [-nompi] [-nocheck]
```

where:

`-p`	This is a "patch" evolution. Bypass the execution of `epx_ftp_getfiles` when the procedure is invoked manually to start a patched evolution
`-noqw`	Skip buil-up and evolution of the QuickWin version.
`-nompi`	Skip buil-up and evolution of the MPI version.
`-nocheck`	Skip buil-up and evolution of the `-check` version.

10.16 Compiler, OS (Windows 7) and hardware upgrade (June 2010)

In June 2010 a new 32-bit PC (DELL Optiplex 740) was installed to replace the former \\sm29 machine. The new machine has an AMD Athlon dual core processor 5400B @ 2.8 GHz, and 4 GB of RAM (of which 3.25 usable).

The operating system installed on this new machine is MS-Windows 7 Professional. The compilation tools installed are:

- MS Visual Studio 2008 Professional Edition (version 9.0.21022.8) with MS .NET Framework Version 3.5 SP1
- Intel Fortran compiler 32bit, Version 11.1.060, Build 20100203.

The same Intel Fortran compiler is then installed also on all other platforms.

10.16.1 Intel 11.1 Fortran compiler

The new Intel Fortran compiler 32bit, Version 11.1.060, Build 20100203 is installed on all platforms. This version of the compiler is the same use now used at CEA (under Linux), and is believed to solve the optimization issues that were previously experienced under the 64-bit operating system with the previous (10.x and 11.0) versions of the Intel compiler.

The strategy adopted for the compiler upgrade is as follows:

- Install and test under the 64-bit platform (\\sm32).
- Then, install under the EUROPLEXUS server (\\sm47).
- Finally, install on the local 32-bit machines (\\sm29 etc.)

10.16.1.1 64-bit compiler installation

The compilation tools available on the 64-bit machine (\\sm32) are:

- MS Visual Studio 2005 Professional Edition (version 8.0.50727-7600) with MS .NET Framework Version 2.0.50727 SP2
- Intel Fortran compiler 64bit, Version 11.1.060, Build 20100203.

To compile under 64-bit, the following modifications are performed:

- The `epx_setvars.bat` script is modified, by replacing the following lines:

```
SET IFV=C:\Program Files (x86)\Intel\Compiler\Fortran\10.1.014\em64t\Bin\ifortvars.bat

if exist "%IFV%" call "%IFV%"
```

by:

```
SET IFV=C:\Program Files (x86)\Intel\Compiler\11.1\060\Bin\ifortvars.bat

if exist "%IFV%" call "%IFV%" intel64
```

- The `epx_filter.f` and the `ordo.f` utility programs are re-compiled under the 64-bit operating system (thus obtaining `epx_filter.exe` and `ordo.exe`):

```
epx_setvars
ifort epx_filter.f
ifort ordo.f
```

Recompilation of libraries

Then, we need to produce new versions of the BLAS/LAPACK and SPLIB libraries (`Libblas.lib` `Liblapack.lib` and `Libsp.lib`, respectively):

- The BLAS/LAPACK libraries are obtained via the following `make_all.pl` script executed in the `Blas_lapack_source` directory:

```
#
# Before launching this script, execute epx_setvars !!!
#
system ("cp lapack-lite-3.1.1.gz ORI.gz");
system ("gunzip lapack-lite-3.1.1.gz");
system ("mv lapack-lite-3.1.1 lapack-lite-3.1.1.tar");
system ("tar xvf lapack-lite-3.1.1.tar");
#
system ("mkdir Liblapack");
system ("cp lapack-lite-3.1.1\\SRC\\*.f Liblapack");
system ("cp lapack-lite-3.1.1\\INSTALL\\dlamch.f Liblapack");
system ("cp lapack-lite-3.1.1\\INSTALL\\slamch.f Liblapack");
system ("rm Liblapack\\xerbla.f");
system ("ifort -c /optimize:5 Liblapack\\*.f");
system ("lib /OUT:Liblapack.lib *.obj");
system ("del *.obj");
#
system ("mkdir Libblas");
system ("cp lapack-lite-3.1.1\\BLAS\\SRC\\*.f Libblas");
system ("ifort -c /optimize:5 Libblas\\*.f");
system ("lib /OUT:Libblas.lib *.obj");
system ("del *.obj");
#
#system ("del ..\\library\\Libblas.lib");
system ("rm -rf lapack-lite-3.1.1");
system ("rm -f lapack-lite-3.1.1.tar");
system ("mv ORI.gz lapack-lite-3.1.1.gz");
#system ("move Libblas.lib ..\\library");
exit;
```

- The SPLIB library is obtained via the following `make_all.pl` script executed in the `Splib_source` directory:

```
system ("del *.obj");
system ("ifort /c /optimize:5 *.f");
system ("del ..\\library\\Libsp.lib");
system ("lib /OUT:Libsp.lib *.obj");
system ("del *.obj");
system ("move Libsp.lib ..\\library");
exit;
```

Note that both above libraries are compiled with the maximum possible optimization (`/optimize:5`).

GLUT library

This library is entirely written in C and therefore would not need to be recompiled due to the Fortran compiler upgrade. Nevertheless, for completeness the library is re-generated as follows:

- Compile all files and produce the library `Glut32.lib`, e.g. by first invoking the `epx_setvars` script (to set the environment), and then by using the following perl script (`make_all.pl`):

```
system ("del *.obj");
system ("del Glut32.lib");
system ("cl /c /o2 *.c");
system ("lib /OUT:Glut32.lib *.obj");
system ("del *.obj");
exit;
```

- Install the library by manually copying the file `Glut32.lib` to the following directory, where also the other 64-bit libraries are believed to reside (see "Standard libraries" above):

```
C:\Program Files (x86)\Microsoft Visual Studio 8\Vc\PlatformSDK\Lib\AMD64
```

F90gl library

The F90GL library must be recompiled. In order to compile and build up the standard version, first invoke the `epx_setvars` command to set the environment and then just type:

```
mf8nio
```

This will produce the following library files:

- `f90gl-1.2.8\lib\f90GL.lib`
- `f90gl-1.2.8\lib\f90GLU.lib`
- `f90gl-1.2.8\lib\f90GLUT.lib`

Copy these files to the Intel compiler's `LIB\Intel64` directory (on the author's machine, this is `C:\Program Files (x86)\ Intel\Compiler\11.1\060\Lib\intel64`).

The following eight Fortran module files are also produced:

- `f90gl-1.2.8\lib\opengl_fwrap.mod`
- `f90gl-1.2.8\lib\opengl_gl.mod`
- `f90gl-1.2.8\lib\opengl_glinterfaces.mod`
- `f90gl-1.2.8\lib\opengl_glu.mod`
- `f90gl-1.2.8\lib\opengl_gluinterfaces.mod`
- `f90gl-1.2.8\lib\opengl_glut.mod`
- `f90gl-1.2.8\lib\opengl_glutinterfaces.mod`
- `f90gl-1.2.8\lib\opengl_kinds.mod`

Copy these files to the Intel compiler's `Include\Intel64` directory (on the author's machine, this is `C:\Program Files (x86)\Intel\Compiler\11.1\060\Include\Intel64`).

Bmplib library

This library is entirely written in C and therefore would not need to be recompiled due to the Fortran compiler upgrade. Nevertheless, for completeness the library is re-generated as follows.

Use a script similar to the ones used for the other two libraries. First invoke the `epx_setvars` command to set the environment (this sets for 64-bit compilation) and then and then use the following perl script (`make_all.pl`):

```
system ("del *.obj");
system ("del Bmplib.lib");
system ("cl /c /o2 *.c");
system ("lib /OUT:Bmplib.lib *.obj");
system ("del *.obj");
exit;
```

Copy the file `Bmplib.lib` to the Intel compiler's `Lib\Ia32` directory (on the author's machine, this is `C:\Program Files (x86)\Intel\Compiler\11.1\060\Lib\Intel64`).

Compiling and linking

The code is completely re-compiled and linked without problems under the 64-bit machine. The command used for compilation is:

`epx_cmp -o` (or equivalently `epx_cmp -5`)

and for the link:

```
epx_lk -o
```

10.16.1.2 32-bit compiler installation (EUROPLEXUS server)

The compilation tools available on the 32-bit "EUROPLEXUS server" machine (\\sm47) are:

- MS Visual Studio 2005 Professional Edition (version 8.0.50727.42) with MS .NET Framework Version 2.0.50727 SP1
- Intel Fortran compiler 64bit, Version 11.1.060, Build 20100203.

To compile under 32-bit, the following modifications are performed:

- The `epx_setvars.bat` script is modified, by replacing the following lines:

```
SET IFV=C:\Program Files\Intel\Compiler\Fortran\10.0.025\IA32\Bin\ifortvars.bat

if exist "%IFV%" call "%IFV%"
```

by:

```
SET IFV=C:\Program Files\Intel\Compiler\11.1\060\Bin\ifortvars.bat

if exist "%IFV%" call "%IFV%" ia32
```

The `epx_filter.f` and the `ordo.f` utility programs are re-compiled under the 32-bit operating system (thus obtaining `epx_filter.exe` and `ordo.exe`):

```
epx_setvars
ifort epx_filter.f
ifort ordo.f
```

Recompilation of libraries

Then, we need to produce new versions of the BLAS/LAPACK and SPLIB This is done exactly like for the 64-bit version as described in Section 10.16.1.1.

GLUT library

This library is entirely written in C and therefore would not need to be recompiled due to the Fortran compiler upgrade. Nevertheless, for completeness the library is re-generated as follows:

- Compile all files and produce the library `Glut32.lib`, e.g. by first invoking the `epx_setvars` script (to set the environment), and then by using the following perl script (`make_all.pl`):

```
system ("del *.obj");
system ("del Glut32.lib");
system ("cl /c /o2 *.c");
system ("lib /OUT:Glut32.lib *.obj");
system ("del *.obj");
exit;
```

- Install the library by manually copying the file `Glut32.lib` to the following directory, where also the other 32-bit libraries are believed to reside (see "Standard libraries" above):

```
C:\Program Files\Microsoft Visual Studio 8\Vc\PlatformSDK\Lib
```

F90gl library

The F90GL library must be recompiled. In order to compile and build up the standard version, first invoke the `epx_setvars` command to set the environment and then just type:

```
mf8nio
```

This will produce the following library files:

- `f90gl-1.2.8\lib\f90GL.lib`
- `f90gl-1.2.8\lib\f90GLU.lib`
- `f90gl-1.2.8\lib\f90GLUT.lib`

Copy these files to the Intel compiler's `LIB\Ia32` directory (on the author's machine, this is `C:\Program Files\Intel\Compiler\11.1\060\Lib\Ia32`).

The following eight Fortran module files are also produced:

- `f90gl-1.2.8\lib\opengl_fwrap.mod`
- `f90gl-1.2.8\lib\opengl_gl.mod`
- `f90gl-1.2.8\lib\opengl_glinterfaces.mod`
- `f90gl-1.2.8\lib\opengl_glu.mod`
- `f90gl-1.2.8\lib\opengl_gluinterfaces.mod`
- `f90gl-1.2.8\lib\opengl_glut.mod`
- `f90gl-1.2.8\lib\opengl_glutinterfaces.mod`
- `f90gl-1.2.8\lib\opengl_kinds.mod`

Copy these files to the Intel compiler's `Include\Ia32` directory (on the author's machine, this is `C:\Program Files\Intel\Compiler\11.1\060\Include\Ia32`).

Bmplib library

This library is entirely written in C and therefore would not need to be recompiled due to the Fortran compiler upgrade. Nevertheless, for completeness the library is re-generated as follows.

Use a script similar to the ones used for the other two libraries. First invoke the `epx_setvars` command to set the environment (this sets for 32-bit compilation) and then and then use the following perl script (`make_all.pl`):

```
system ("del *.obj");
system ("del Bmplib.lib");
system ("cl /c /o2 *.c");
system ("lib /OUT:Bmplib.lib *.obj");
system ("del *.obj");
exit;
```

Copy the file `Bmplib.lib` to the Intel compiler's `Lib\Ia32` directory (on the author's machine, this is `C:\Program Files\Intel\Compiler\11.1\060\Lib\Ia32`).

Compiling and linking

The code is completely re-compiled and linked without problems under the 32-bit machine. The command used for compilation is:

```
epx_cmp -o
```

and for the link:

```
epx_lk -o
```

10.16.1.3 32-bit compiler installation (Windows 7 machine)

The compilation tools available on the 32-bit Windows 7 machine (\\sm58) are:

- MS Visual Studio 2008 Professional Edition (version 9.0.21022.8) with MS .NET Framework Version 3.5 SP1
- Intel Fortran compiler 32bit, Version 11.1.060, Build 20100203.

To set up a local working Fortran compiling environment, similar customizations to those described above for the 32-bit EUROPLEXUS Server machine must be applied. The compiler versions are the same, so the library files produced for the other machine may be used directly. However, the versions of MS Visual studio are different (2008 instead of 2005), and so the locations where the library and include files related to the Studio environment are to be placed may differ.

OpenGL-related include files

These include files are copied from the EUROPLEXUS server machine (MS Visual Studio 2005):

```
\\sm47\C\Program Files\Microsoft Visual Studio 8\Vc\PlatformSDK\Include\gl\glut.h
\\sm47\C\Program Files\Microsoft Visual Studio 8\Vc\PlatformSDK\Include\gl\glutf90.h
\\sm47\C\Program Files\Microsoft Visual Studio 8\Vc\PlatformSDK\Include\gl\glaux.h (if needed)
```

to the local machine (MS Visual Studio 2008):

```
C:\Program Files\MicrosoftSDKs\Windows\v6.0A\Include\gl\glut.h
C:\Program Files\MicrosoftSDKs\Windows\v6.0A\Include\gl\glutf90.h
C:\Program Files\MicrosoftSDKs\Windows\v6.0A\Include\gl\glaux.h (if needed)
```

GLUT library

This library is copied from the EUROPLEXUS server machine (MS Visual Studio 2005):

```
\\sm47\C\Program Files\Microsoft Visual Studio 8\Vc\PlatformSDK\Lib\Glut32.lib
```

to the local machine (MS Visual Studio 2008):

```
C:\Program Files\MicrosoftSDKs\Windows\v6.0A\Lib
```

F90gl library

The F90GL library files are copied from the EUROPLEXUS server machine:

```
\\sm47\C\Program Files\Intel\Compiler\11.1\060\Lib\Ia32\f90GL.lib
\\sm47\C\Program Files\Intel\Compiler\11.1\060\Lib\Ia32\f90GLU.lib
\\sm47\C\Program Files\Intel\Compiler\11.1\060\Lib\Ia32\f90GLUT.lib
```

to the local machine:

```
C:\Program Files\Intel\Compiler\11.1\060\Lib\Ia32\f90GL.lib
C:\Program Files\Intel\Compiler\11.1\060\Lib\Ia32\f90GLU.lib
C:\Program Files\Intel\Compiler\11.1\060\Lib\Ia32\f90GLUT.lib
```

The F90GL module files are copied from the EUROPLEXUS server machine:

```
\\sm47\C\Program Files\Intel\Compiler\11.1\060\Include\Ia32\opengl_fwrap.mod
\\sm47\C\Program Files\Intel\Compiler\11.1\060\Include\Ia32\opengl_gl.mod
\\sm47\C\Program Files\Intel\Compiler\11.1\060\Include\Ia32\opengl_glinterfaces.mod
\\sm47\C\Program Files\Intel\Compiler\11.1\060\Include\Ia32\opengl_glu.mod
\\sm47\C\Program Files\Intel\Compiler\11.1\060\Include\Ia32\opengl_gluinterfaces.mod
\\sm47\C\Program Files\Intel\Compiler\11.1\060\Include\Ia32\opengl_glut.mod
\\sm47\C\Program Files\Intel\Compiler\11.1\060\Include\Ia32\opengl_glutinterfaces.mod
\\sm47\C\Program Files\Intel\Compiler\11.1\060\Include\Ia32\opengl_kinds.mod
```

to the local machine:

```
C:\Program Files\Intel\Compiler\11.1\060\Include\Ia32\opengl_fwrap.mod
C:\Program Files\Intel\Compiler\11.1\060\Include\Ia32\opengl_gl.mod
C:\Program Files\Intel\Compiler\11.1\060\Include\Ia32\opengl_glinterfaces.mod
C:\Program Files\Intel\Compiler\11.1\060\Include\Ia32\opengl_glu.mod
C:\Program Files\Intel\Compiler\11.1\060\Include\Ia32\opengl_gluinterfaces.mod
C:\Program Files\Intel\Compiler\11.1\060\Include\Ia32\opengl_glut.mod
C:\Program Files\Intel\Compiler\11.1\060\Include\Ia32\opengl_glutinterfaces.mod
C:\Program Files\Intel\Compiler\11.1\060\Include\Ia32\opengl_kinds.mod
```

Bmplib library

The Bmplib library files are copied from the EUROPLEXUS server machine:

```
\\sm47\C\Program Files\Intel\Compiler\11.1\060\Lib\Ia32\bmplib.lib
```

to the local machine:

```
C:\Program Files\Intel\Compiler\11.1\060\Lib\Ia32\bmplib.lib
```

10.16.2 Compilation times on the various platforms

The compilation times (in CPU minutes) obtained for the various compilation modes and on the various platforms by means of the new Intel Fortran compiler version 11.1.060 are reported in the following table:

Platform/compilation	SM58 (W7)	SM47 (old server)	SM32 (64-bit)	SM29 (ex Folco)
epx_cmp -o	52	48	29	34
epx_cmp –o -c	144	262	101	118
epx_cmp –o m_domaine	1	1	1	1
epx_cmp –o –c m_domaine	27	~60	9	17

The above results concern either a full compilation of the EUROPLEXUS code, or the compilation of the slower routine to compile (m_domaine.ff).

Here are the machine characteristics:

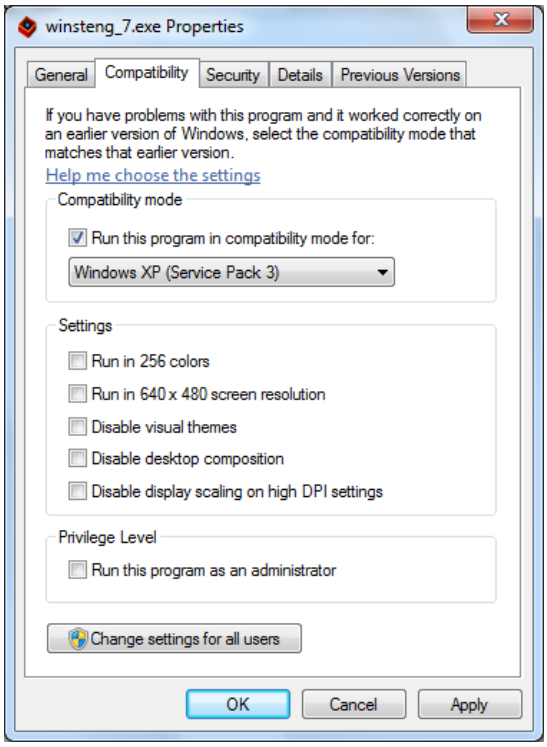
Machine	OS	Processor	RAM
SM58	Windows 7	AMD Athlon Dual Core 5400 B @ 2.8 GHz	4 GB

SM47	n/a	n/a	n/a
SM32	XP 64-bit SP2	Intel XEON 5110 @ 1.60 GHz, 1.60 GHz	32 GB
SM29	XP SP3	Intel XEON 2 CPU @ 2.80 GXz	1 GB

10.16.3 Adobe PostScript Driver installation under Windows 7

The Adobe PostScript Driver installation package (`winsteng.exe`) is not compatible with Windows 7: if one runs the installer program as such one gets a "severe error". To avoid this problem, do the following:

- Right-click on the `winsteng.exe` file and select Properties.
- Set Compatibility Mode to Windows XP SP3:



- Apply
- Close the Properties window, then launch the `winsteng.exe` program by double-clicking on it and perform the installation as usual (see e.g. Section 10.7.13).

10.16.4 MiKTeX under Windows 7

An attempt was made to install the previous version of MiKTeX ("small" MiKTeX 2.4, August 2005) and run it under Windows 7 on the \\sm58 machine. Installation proceeds smoothly, but then the program does not work correctly. When compiling by LaTeX a document, a message appears indicating that files are older than 5 years and an upgrade is necessary.

Even by replacing the installed directory (`C:\Texmf`) by the directory from another (working) machine, the program does not work.

The reason is believed to be that this version of MiKTeX is not compatible with the Windows 7 operating system (this seems indeed to be the case looking at information on the MiKTeX web site at www.miktex.org).

Then, an attempt was made to use the latest version of MiKTeX from the web site (version 2.8). The "small" MiKTeX installer (`basic-miktex-2.8.3761.exe`) was downloaded and then executed to perform the installation, which went smoothly. During installation, remember choose to "install missing packages on the fly".

Just after installing, perform a reboot of the machine (not requested by the installer) otherwise the MiKTeX 2.8 menu is not visible under Start → Programs!

However, one notes that the installation of the new version differs from the old one. Instead of installing into "`C:\Texmf`" it installs into "`C:\Program Files\MiKTeX2.8`" and the organization of sub-directories is different. In particular, it is not evident where to put the customized files (for Hevea etc.) in the new organization. In fact, when running LaTeX to build up an EUROPLEXUS man page, the code stops because it cannot find the file `hevea.sty`.

Installing personal style files

From the new MiKTeX documentation, the recommended way of letting personal style files be available to LaTeX is to register a user-managed TEXMF directory. This is done as follows:

- Create a user-managed directory (say `C:\Texmf` : I use this name because it reminds the place where the files were put in the older installations)
- Place personal style files in this directory, by imitating the directory tree in the MiKTeX installation directory. For LaTeX style files (`.sty`) and with the above choice of the local directory name, this means that the files must go into `C:\texmf\tex\latex\folco`
- As administrator: use MiKTeX Options to register the `C:\Texmf` directory. Click Start → Programs → MiKTeX 2.8 → Maintenance (Admin) → Settings (Admin) to open the MiKTeX Options Window. Click on the Roots tab. The Roots page shows the list of currently registered root directories. Click Add. In the following dialog box browse to `C:\Texmf` and click OK. The new directory is appended to the list.

After the above operations, MiKTeX is able to locate the personal style files.

Extending the MiKTeX input search path

Another problem to be solved under the new version of MiKTeX is how to extend the MiKTeX input search path (see e.g. Section 7.5.10). The previous configuration file miktex.ini is no longer available. This can be done by setting the environment variable `TEXINPUTS` as follows:

```
set TEXINPUTS=\\sm47\e\europlexus\manual_filtered
```

In this way the MiKTeX products (LaTeX, pdfTeX etc.) search also in this directory in order to locate files to be included in the compilation like in the following example:

```
include{gb_abc.tex}
```

The file gb_abc.tex would not be found if the `TEXINPUTS` variable is not set correctly. The varisble must be set as a system variable, so that it is set for all users.

Installing Tth

To install `tth` (see Section 10.5.5) place it under the `C:\Texmf\doc\tth` (the doc files) and `C:\Texmf\miktex\bin` (the executable) like in the previous installations (make sure that this directory is on the path!). Provided `C:\Texmf` has been registered as explained above, the files will be found upon execution of the manual generation scripts.

Installing Hevea

To install `hevea` (see Section 10.5.4) place it uder the `C:\hevea` directory like in the previous installations. The `hevea.sty` style file is anyway located in the `C:\texmf\tex\latex\folco` directory so the file is found when necessary.

10.16.5 New “EUROPLEXUS server” (\\sm47)

As results from the table in Section 10.16.2, compilation with the new Fortran compiler 11.1 has become much longer in terms of CPU time, especially on the older machine (\\sm47). This machine is the current “EUROPLEXUS server” and has to perform a complete evolution (32-bit version) every night. Evolutions (with modules, which means full code recompilation) takes about 9 hours to produce the 4 32-bit versions: optimized, QuickWin, MPI and -check.

Therefore, it is decided to replace the 32-bit EUROPLEXUS server. The old \\sm47 machine is set aside and replaced by the \\sm29 machine (renamed \\sm47 so that no changes are needed in the procedures).

The previous software is installed on the new server, with some exceptions, in particular:

- MS Visual Studio 2008 Professional Edition (version 9.0.21022.8) with MS .NET Framework Version 3.5 SP1
- Intel Fortran compiler 32bit, Version 11.1.060, Build 20100203.
- MiKTeX 2.8. This is same version as for the Windows 7 machine, as described in Section 10.16.4. An attempt to re-install the older version 2.4 failed, with exactly the same symptoms mentioned before for the Windows 7 platform. Therefore, also for uniformity, it was decided to switch to the new version (now on all platforms). Recall that this necessitates the setting of an additional system environment variable `TEXINPUTS=E:\EUROPLEXUS\manual_filtered`, in ordet to extend the search for LaTeX input files.

In order to make it possible to automatically upload the executables and the manuals to the Consortium web site from the new server during the evolution process, the “initialization” operations relative to the error message “The server’s host key is not cached in the registry” described in Section 10.14.20 are executed on the new server.

10.16.6 Automatic upload of all executables

All executables produced during evolutions are now automatically uploaded to the Consortium web site. These are summarized in the following Table:

Platform	Version	Name
32-bit	Normal	europlexus.exe.gz
32-bit	QuickWin	europlexusqw.exe.gz
32-bit	MPI	europlexus_mpi.exe.gz
32-bit	-check	europlexus_check.exe.gz
64-bit	Normal	europlexus64.exe.gz
64-bit	MPI	europlexus64_mpi.exe.gz

The evolution procedures `epx_evol_start.pl` (32-bit) and `epx_evol_64.pl` (64-bit) are updated accordingly. A new procedure `epx_ftp_putexe_64.pl` is produced for the transfer of the 64-bit executables.

10.17 Hardware upgrade to 64-bit Windows 7 (October 2010)

In October 2010 a new 64-bit PC (DELL Optiplex 780) \\sm12 was installed to replace the former \\sm58 (Dell Optiplex 740) machine. The new machine has an Intel Core2 Quad processor Q9550 @ 2.83 GHz, and 8 GB of RAM (all usable under the 64-bit operating system installed).

The operating system installed on this new machine is MS-Windows 7 Professional (64-bit version). The compilation tools installed are:

- MS Visual Studio 2008 Professional Edition (version 9.0.21022.8) with MS .NET Framework Version 3.5 SP1
- Intel Fortran compiler 64bit, Version 11.1.067, Build 20100806.

10.17.1 Situation prior to the upgrade

The available machines and their characteristics are as follows:

\\sm47 ("Old" server)
Architecture: 32-bit
O-S: Windows XP Professional SP3
Visual Studio: 2008 V 9.0.21022.8, .NET Framework 3.5 SP1
Fortran Compiler: Intel 11.1.060 32-bit, Build 20100203

\\sm58 ("Old" Folco's machine)
Architecture: 32-bit
O-S: Windows 7 Professional
Visual Studio: 2008 V 9.0.21022.8, .NET Framework 3.5 SP1
Fortran Compiler: Intel 11.1.060 32-bit, Build 20100203

\\sm32 ("Old" 64-bit server)
Architecture: 64-bit
O-S: Windows XP Professional x64 SP2
Visual Studio: 2005 V 8.0.50727.762, .NET Framework 2.0.50727 SP1
Fortran Compiler: Intel 11.1.060 64-bit, Build 20100203

\\sm12 ("New" 64-bit Folco's machine)
Architecture: 64-bit
O-S: Windows 7 Professional
Visual Studio: 2008 V 9.0.21022.8, .NET Framework 3.5 SP1
Fortran Compiler: Intel 11.1.067 64-bit, Build 20100806

The scope here is to remove the "Old" EUROPLEXUS server (\\sm47) and replace it by the "Old" Folco's machine (\\sm58, which will be renamed \\sm47 for convenience). At the same time, Folco will receive a new machine (64-bit instead of 32-bit), namely \\sm12.

10.17.2 Situation after the upgrade

The situation after the upgrade will then be:

\\sm47 ("New" server, was previously named \\sm58)
Architecture: 32-bit
O-S: Windows 7 Professional
Visual Studio: 2008 V 9.0.21022.8, .NET Framework 3.5 SP1
Fortran Compiler: Intel 11.1.060 32-bit, Build 20100203

\\sm32 ("Old" 64-bit server)
Architecture: 64-bit
O-S: Windows XP Professional x64 SP2
Visual Studio: 2005 V 8.0.50727.762, .NET Framework 2.0.50727 SP2
Fortran Compiler: Intel 11.1.060 64-bit, Build 20100203

\\sm12 ("New" 64-bit Folco's machine)
Architecture: 64-bit
O-S: Windows 7 Professional
Visual Studio: 2008 V 9.0.21022.8, .NET Framework 3.5 SP1
Fortran Compiler: Intel 11.1.067 64-bit, Build 20100806

10.17.3 Fixing the epx_setvars script

Until now, two versions of the `epx_setvars` script were available, one for the 32-bit environment (from the \\sm47 machine) and one for the 64-bit environment (from the \\sm32 machine). An effort has been made to write down the script in a more portable way, so that the same version works on all platforms. The new script is listed hereafter:

```
@echo off

rem This version of the epx_setvars script should be reasonably platform-independent.
rem It requires INTEL the Fortran Compiler 11 to be installed (environment variable
rem IFORT_COMPILER11 should be set).
rem It should automatically detect whether the machine is 32 or 64 bit by using
rem the environment variable PROCESSOR_ARCHITECTURE (which should be set
rem either to x86 or to AMD64.

if "%EPX_VARS_SET%" == "OK" goto End

echo Setting the variables ...

rem I am obliged to comment out the following safety tests because they
rem fail with 64-bit compilers (although they would work OK on 32-bit)
rem due to the fact that under 64-bit the directory name in the environment
rem variable IFORT_COMPLIER11 is
rem    C:\Program Files (x86)\ ...
rem The spaces in the directory name do not cause any problem, but the
rem parentheses (especially the closed parenthesis) do!!!
rem If the file name contains a parenthesis, then the exist command
rem does not work. The call command works if I put a double set of "
rem around the file name (see below), but that does not help with
rem the exists command ...
rem To solve the problem with parantheses, one could try to escape them
rem by the ^ character, but it seems too complicated and I did not try it.

rem IF exist "%IFORT_COMPILER11%" (

  SET IFV="%IFORT_COMPILER11%Bin\ifortvars.bat"

rem ) ELSE (
rem echo ERROR : directory IFORT_COMPILER11 = '%IFORT_COMPILER11%' does not exist !
rem exit /B 1
rem )

rem IF not exist "%IFV%" (
rem   echo ERROR : ifortvars.bat script does not exist !
rem   exit /B 1
rem )

IF "%PROCESSOR_ARCHITECTURE%" == "x86" (
  @call ""%IFV%"" ia32
  SET EPX_VARS_SET=OK
  exit /B 0
)

IF "%PROCESSOR_ARCHITECTURE%" == "AMD64" (
  @call ""%IFV%"" intel64
  SET EPX_VARS_SET=OK
  exit /B 0
)

echo ERROR : wrong value of environment variable PROCESSOR_ARCHITECTURE =
'%PROCESSOR_ARCHITECTURE%' !
echo         accepted values are : x86 or AMD64
exit /B 1

:End
```

Unfortunately, a problem occurs for the 64-bit compilers (see comments in the script above), because the `IFORT_COMPILER11` environment variable for this architecture is set to:

```
C:\Program Files (x86)\ ...
```

The spaces in the directory name do not cause problems, but the parentheses in `(x86)` do! The `exists` command fails when the file name contains a parenthesis. A possible workaround might be to

escape the parenthesis with a caret sign ^, but this seems complicated and I have not tried it. For this (stupid) reason, I may not activate the checks, which use the `exists` command. The `call` command also suffers with the parentheses in the directory name, but here it suffices to put two sets of " signs around the file name rather than one (but this trick does not work for the `exists` command).

Although not optimal, the above version of the script now works for all the platforms we have. NO !!! The compilation of some files (e.g. M_RENDER_CONSTANTS) fails on \\sm32 (but only when launched from the evolution procedure, while when launched "by hand" in a console window it works!) because the includes `opengl_gl.mod` etc. cannot be found! For the moment therefore I restore the old `epx_setvars.bat` on the \\sm32 machine.

One possible way of solving the problems mentioned above with the batch command language is to re-write the `epx_setvars` script in Perl (and the transform it in batch by the `pl2bat` utility) rather than in batch language directly. This is underway at the moment.

10.17.4 Installing the Intel 11.1.067 Fortran Compiler

To complete installation of the Intel 11.1.067 Fortran Compiler, the following steps were performed:

OpenGL-related include files

These include files are copied from the EUROPLEXUS 64-bit server machine (MS Visual Studio 2005):

```
\\sm32\C\Program Files (x86)\Microsoft Visual Studio 8\Vc\PlatformSDK\Include\gl\glut.h
\\sm32\C\Program Files (x86)\Microsoft Visual Studio 8\Vc\PlatformSDK\Include\gl\glutf90.h
\\sm32\C\Program Files (x86)\Microsoft Visual Studio 8\Vc\PlatformSDK\Include\gl\glaux.h
```

to the local machine (MS Visual Studio 2008):

```
\\sm12\C\Program Files\MicrosoftSDKs\Windows\v6.0A\Include\gl\glut.h
\\sm12\C\Program Files\MicrosoftSDKs\Windows\v6.0A\Include\gl\glutf90.h
\\sm12\C\Program Files\MicrosoftSDKs\Windows\v6.0A\Include\gl\glaux.h
```

GLUT library

This library is copied from the EUROPLEXUS 64-bit server machine (MS Visual Studio 2005):

```
\\sm32\C\Program Files (x86)\Microsoft Visual Studio 8\Vc\PlatformSDK\Lib\AMD64\Glut32.lib
```

to the local machine (MS Visual Studio 2008):

```
\\sm12\C\Program Files (x86)\Microsoft SDKs\Windows\v6.0A\Lib\x64\Glut32.lib
```

F90gl library

The F90GL library files are copied from the EUROPLEXUS 64-bit server machine:

```
\\sm32\C\Program Files (x86)\Intel\Compiler\11.1\060\Lib\intel64\f90GL.lib
\\sm32\C\Program Files (x86)\Intel\Compiler\11.1\060\Lib\intel64\f90GLU.lib
\\sm32\C\Program Files (x86)\Intel\Compiler\11.1\060\Lib\intel64\f90GLUT.lib
```

to the local machine:

```
\\sm12\C\Program Files (x86)\Intel\Compiler\11.1\067\Lib\intel64\f90GL.lib
\\sm12\C\Program Files (x86)\Intel\Compiler\11.1\067\Lib\intel64\f90GLU.lib
\\sm12\C\Program Files (x86)\Intel\Compiler\11.1\067\Lib\intel64\f90GLUT.lib
```

The F90GL module files are copied from the EUROPLEXUS 64-bit server machine:

```
\\sm32\C\Program Files (x86)\Intel\Compiler\11.1\060\Include\intel64\opengl_fwrap.mod
\\sm32\C\Program Files (x86)\Intel\Compiler\11.1\060\Include\intel64\opengl_gl.mod
\\sm32\C\Program Files (x86)\Intel\Compiler\11.1\060\Include\intel64\opengl_glinterfaces.mod
\\sm32\C\Program Files (x86)\Intel\Compiler\11.1\060\Include\intel64\opengl_glu.mod
\\sm32\C\Program Files (x86)\Intel\Compiler\11.1\060\Include\intel64\opengl_gluinterfaces.mod
\\sm32\C\Program Files (x86)\Intel\Compiler\11.1\060\Include\intel64\opengl_glut.mod
\\sm32\C\Program Files (x86)\Intel\Compiler\11.1\060\Include\intel64\opengl_glutinterfaces.mod
\\sm32\C\Program Files (x86)\Intel\Compiler\11.1\060\Include\intel64\opengl_kinds.mod
```

to the local machine:

```
\\sm12\C\Program Files (x86)\Intel\Compiler\11.1\067\Include\intel64\opengl_fwrap.mod
\\sm12\C\Program Files (x86)\Intel\Compiler\11.1\067\Include\intel64\opengl_gl.mod
\\sm12\C\Program Files (x86)\Intel\Compiler\11.1\067\Include\intel64\opengl_glinterfaces.mod
\\sm12\C\Program Files (x86)\Intel\Compiler\11.1\067\Include\intel64\opengl_glu.mod
\\sm12\C\Program Files (x86)\Intel\Compiler\11.1\067\Include\intel64\opengl_gluinterfaces.mod
\\sm12\C\Program Files (x86)\Intel\Compiler\11.1\067\Include\intel64\opengl_glut.mod
\\sm12\C\Program Files (x86)\Intel\Compiler\11.1\067\Include\intel64\opengl_glutinterfaces.mod
\\sm12\C\Program Files (x86)\Intel\Compiler\11.1\067\Include\intel64\opengl_kinds.mod
```

Bmplib library

The Bmplib library files are copied from the EUROPLEXUS 64-bit server machine:

```
\\sm32\C\Program Files (x86)\Intel\Compiler\11.1\060\Lib\intel64\bmplib.lib
```

to the local machine:

```
\\sm12\C\Program Files (x86)\Intel\Compiler\11.1\067\Lib\intel64\bmplib.lib
```

With the above settings, the compilation and link of EUROPLEXUS work well from the command line on the new machine \\sm12.

10.17.5 Setting the EUROPLEXUS and other environment variables

On the new \\sm12 machine, the `EUROPLEXUS` environment variable is set to `\\SM32\E\EUROPLEXUS`, and the same is done for the other similar variables `GNUDIR`, `UTILDIR`, `RUNDIR` etc, which also appear in the `PATH`. This means that the new machine uses \\sm32 as "server" rather than \\sm47. This is in accordance with the fact that both \\sm32 and \\sm12 are 64-bit machines, while \\sm47 is a 32-bit machine.

The only environment variable which keeps the old \\SM47 root is `TEXINPUTS` (set to `\\SM47\E\EUROPLEXUS\MANUAL_FILTERED`). This is due to the fact that, at the moment, the \\sm32 machine does not fabricate the EUROPLEXUS manual, and so its MANUAL_FILTERED directory is empty.

10.17.6 Problem with Visual Studio project

A problem occurred with the Visual Studio Project used to debug EUROPLEXUS. By typing the `epx_init -e` command on the \\sm12 machine, the files are taken from \\sm32. Note that the `-e` switch (empty project) is necessary, else an error message occurs. The project converter appears (because the project from \\sm32 is not compatible with \\sm12 at the moment). Following the instructions, the project is successfully converted and the files may then be added "by hand" in the usual way.

However, when building the project, an error occurs (apparently at the end of the filtering procedure and before the compilation). The error message is:

```
1>mt.exe : general error c10100b1: Failed to load file "..\x64\Debug\filter.exe". The
system cannot find the path specified.
1>Build log was saved at "file://c:\TEMP\Test\filter\x64\Debug\BuildLog.htm"
1>filter - 1 error(s), 0 warning(s)
```

It is not clear why the system apparently looks for a file `filter.exe`, which does not exist. However, despite this message, the filtering, the successive compilation and the linkage do work and the resulting executable may be run an debugged as usual.

Therefore, for the moment, this problem is left aside.

10.17.7 Installation of MiKTeX

The MiKTeX package is installed exactly as described in Section 10.16.4. As mentioned above in Section 10.17.5, the `TEXINPUTS` environment variable is left pointing to \\sm47, for the moment. The manual generation works correctly.

10.17.8 Installation of Cast3M

The 64-bit version of Cast3M (suitable for the creation of very large meshes) is installed, by first installing the basic package, and then by replacing the files with those from the \\sm32 machine. The basic package was installed by Loris on the directory `C:\Program Files (x86)\Joint Research Centre\Visual Cast3m`, rather than on the usual one (`C:\Cast3m`), but for the moment at least I leave it as it is (Visual Cast3m works anyway).

For batch execution, the `k2008b.pl` procedure (on \\sm32\e\europlexus\run) is slightly modified, so that it will work for both 64-bit machines (\\sm32 and \\sm12):

The variable:

```
$CastemPath = 'D:\Users\Pegon\Castem2008';
```

is replaced by:

```
$CastemPath = 'C:\Castem2008';
```

The setting:

```
$ENV{ESOPE_TEMP} = "D:\\TEMP";
```

is replaced by:

```
$ENV{ESOPE_TEMP} = "C:\\TEMP";
```

The first setting requires the creation on the `C:` disk (this is the only available disk in the new machines) of directory `C:\castem2008`, into which the following subdirectories are copied from the \\sm32 machine:

- `bin` (this contains omong other things the 64-bit "batch" executable: `prov_b.exe`)
- `dgibi`
- `divers`
- `documents`
- `gibi_files`

The same directory structure is created also on \\sm32, by copying subdirectores from `\\sm32\D\Users\Pegon\Castem2008` (this folder is kept unchanged because it also contains other subdirectories, for the creation of the 64-bit version).

On both 64-bit machines, the 64-bit batch version of Cast3m is then invoked by:

```
K2008b -s <size> myfile
```

where `<size>` is the memory size (typically `4000`, but much larger values can be used if necessary) and `myfile.dgibi` is the name of the input file.

10.17.9 Efficiency measurements

Here are some measured CPU times for the execution of all EUROPLEXUS benchmarks (as of October 2010, ~ 745 benchmarks):

Machine	Type	Description	CPU (minutes)
\\sm32	64-bit XP	64-bit server	46
\\sm47 (old)	32-bit XP	Local execution	64
\\sm81	64-bit W7	Remote execution with \\sm47 old server	87
\\sm81	64-bit W7	Local execution	53
\\sm81	64-bit W7	Remote execution with \\sm58, W7	27

\\sm12	64-bit W7	Remote execution with \\sm32, XP	51

It is interesting to note that the execution on the same machine (\\sm81) passes from 87 minutes to 53 minutes if the benchmarks are first copied on the local directory instead of fetching them from the server one by one. The CPU time then drops to only 27 minutes, when the "server" is a machine with the same operating system (Windows 7 in this case) rather than with another operating system (XP).

As concerns compilation time, the full compilation (optimized) of EUROPLEXUS (~1862 files of which ~339 F90 modules) takes the following CPU times:

Machine	Type	Description	CPU (minutes)
\\sm12	64-bit XP	Remote compilation with \\sm32, XP	24

10.17.10 Installation of PostScript printer driver

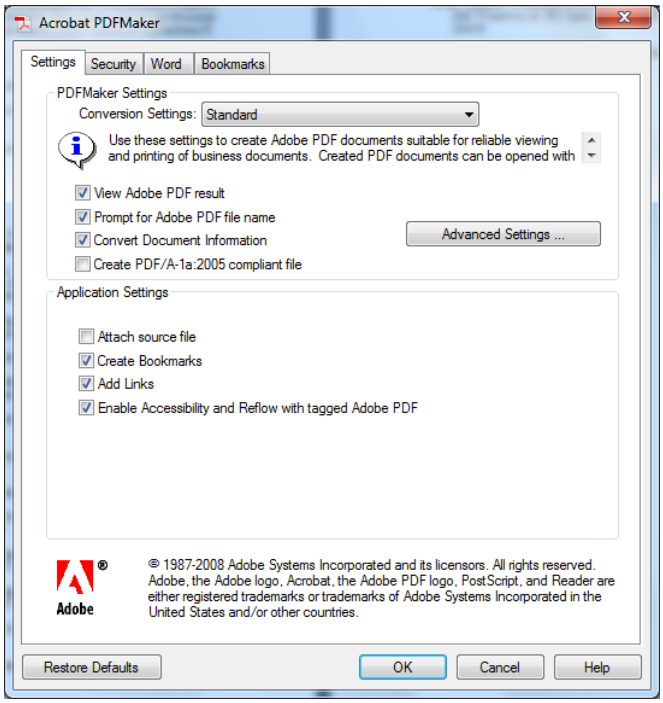
An attempt has been made to install the Adobe PostScript driver as explained in Section 10.16.3 for the 32-bit version of the Windows 7 OS. However, installation fails even by setting Compatibility Mode to XP SP3.

Since some malfunctionings of the driver had been observed also under the 32-bit version of Windows 7 (where installation did work), we look for an alternative way of printing to a PostScript file under Windows 7 64-bit, to be later distilled to PDF, or to print directly to PDF (see following Sections).

10.17.11 Printing to PDF directly from Word

From MS Word 2007 under Windows 7 64-bit one should (in theory) be able to print directly to PDF as follows:

- Locate the Acrobat tab that should appear in the Word main window in the top menu bar (after View and MathType).
- In the Create Adobe PDF group, click on Preferences. The following Acrobat PDFMaker dialog opens up



- The Standard setting is chosen by default. You may want to create a special (folco)setting as explained previously, bli clicking on Advanced Settings, by choosing “No Downsampling” in the Images tab, and by saving the settings as “folco” (it should be necessary to do this only the first time).

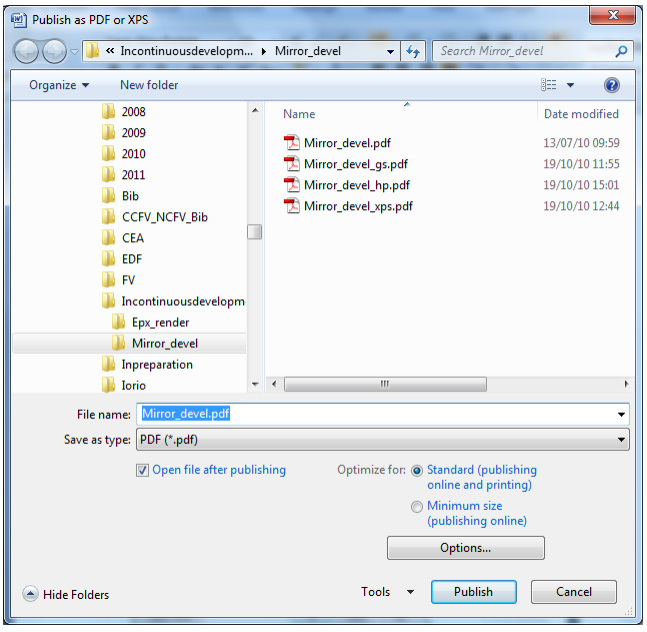
- Choose the folco setting and click OK.

- In the Create Adobe PDF group, click on Generate PDF (?)

The conversion seems extremely slow, then the Word program blocks when I do this! When I re-open Word the Acrobat tab is no longer available!

Another possibility is as follows.

- Click on the Windows icon on the top-left of the Word window and select Save AS → PDF or XPS. The following dialog opens



- Select the desired File name, Save as type, Optimize for and Options, then click on Publish

This method allows less customizations than the other one (distilling options are not available?) but it works relatively fast and, in addition, hyperlinks set in Word (e.g. Table of Contents, cross-references etc.) are retained in the PDF version (which was not the case when distilling manually).

10.17.12 Printing to PDF directly from Internet Explorer

Under Windows 7 32-bit a problem occurred when trying to print to the "Generic Postscript Printer (distiller)" from Internet Explorer. The system illued a file error whatever the name of the chosen file (permissions issue?).

Under the 64-bit cersion of Windows 7, it seems impossible to install the "Generic Postscript Printer (distiller)", so another way must be found to produce PDF from Internet Explorer. Do as follows:

- In Internet Exploree, make sure that the Adobe PDF toolbar is visible. If not, click on View→ Toolbars → Adobe PDF.
- Click on Convert → Preferences and adjust the (very few available) preferences.
- Click on Convert → Convert Web Page to PDF.

10.17.13 Printing from FrameMaker

As is well-known, FrameMaker is extremely sensitive (font issues) to the default printer that is chosen when a FrameMaker document is opened. Changing the printer in an open document may cause a lot of problems. So the preferred way of working so far was to set the "Generic PostScript Printer (distiller)" as the default printer before opening any FrameMaker document.

Since this driver may no longer be installed under Windows 7 64-bit, there is a problem.

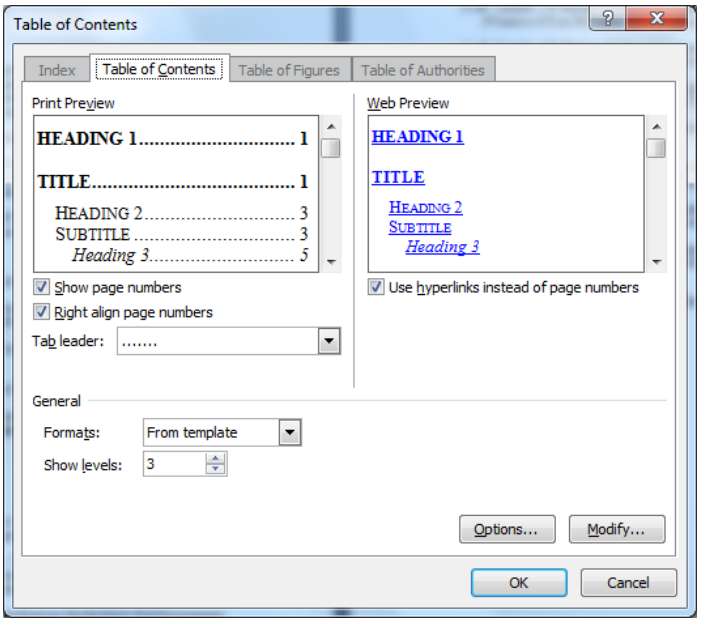
One way of solving the problem, is to install another PostScript driver (just any of the drivers available in the system when trying to install a new printer), then set this printer as default and then printing to this printer with the "Print to file" box checked from Framemaker.

At the moment I have tried installing and using the "HP Color Laserjet 8500 PS" printer. There are some error messages occasionally, but more or less the process seems to work. Maybe it is possible to find a simpler "generic" driver.

10.17.14 Fixing a corrupted Table of Contents in Word

Occasionally the Table of Contents in a Word document (such as this one for example) gets corrupted and it won't update any more. To fix the problem, proceed as follows:

- Remove the corrupted Table of Contents: click and select the whole table, then press Shift DELETE.
- On the References Tab, click on Table of Contents.
- Do not choose any of the Built-in Tables of Contents, but rather click on "Insert Table of Contents". This allows to select the desired paragraph tags to appear in the Table.



- Click on Options and then unselect the undesired tags (such as Title and Subtitle, for example) and select the desired additional ones (SUBNAME, level 3).
- Click OK. The table is generated. Adjust the title "Contents" if necessary.
- Now the Table should update correctly.

10.17.15 Installing Gvim 6.3 under 64-bit

Installation of gvim 6.3 under 64-bit Windows 7 proceeds apparently smoothly as usual, but the result is that the right-click context menu "Edit with Vim" in the Windows Explorer application is not installed correctly.

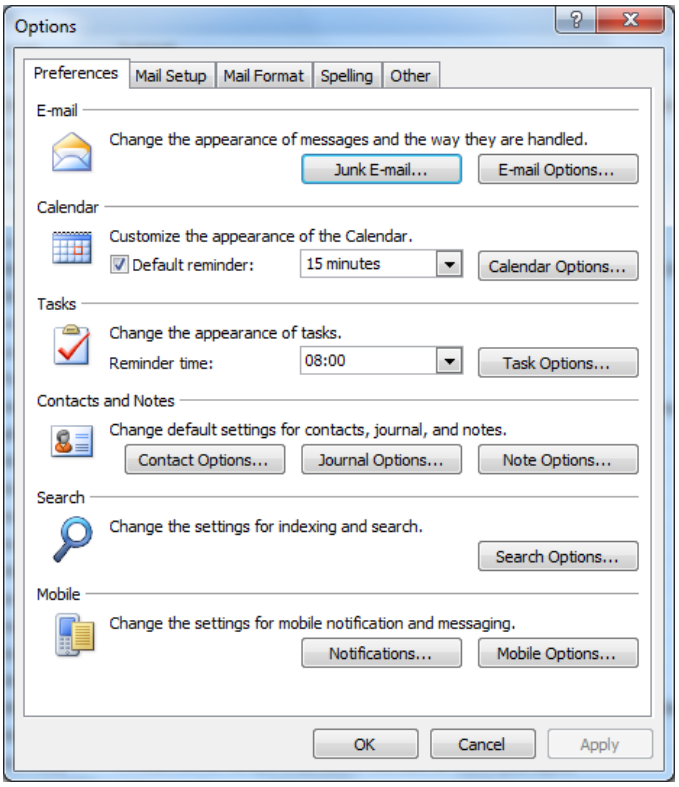
To add this contexyt menu by hand (after having installed Gvim 6.3), proceed as follows:

- Log in as administrator.
- Launch the `regedit` application (enter `regedit` in the Start menu, "search" dialog and when `regedit.exe` appears double-click on it.
- Open the `HKEY_CLASSES_ROOT\*\shell` directory (note that the * here is a literal star appearing in the directory name, it does not stand for "any file").
- In this directory, create a new key (sub-directory) named `Edit with Vim`.
- In the `Edit with Vim` directory, create a new key (sub-directory) named `command`.
- In the command directory, edit the `(Default)` value, so that it contains the following string (including the four double apices): `"C:\Vim\vim63\gvim.exe" "%1"`.
- Exit `regedit`.

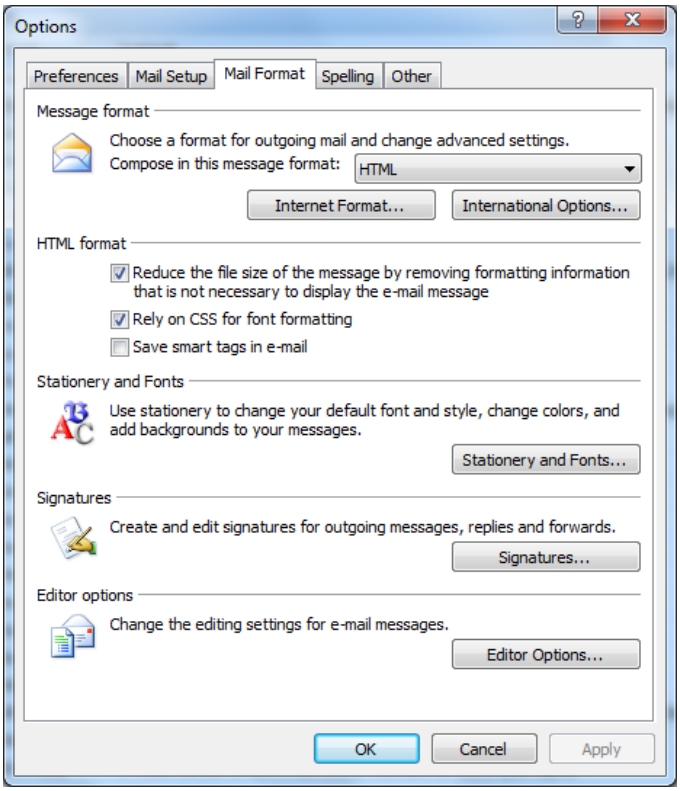
Now the "edit with Vim" context menu should be available for any file in Windows Explorer.

10.17.16 Creating an automatic signature under MS Outlook 2007

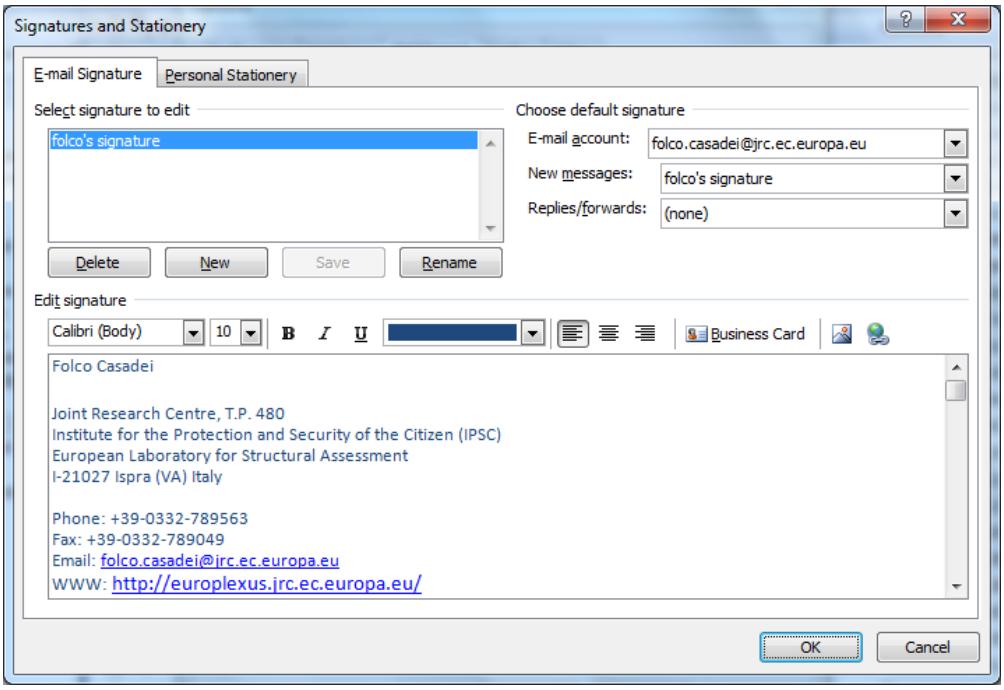
To create an automatic signature under MS Outlook 2007, clich on Tools → Options. The following dialog appears:



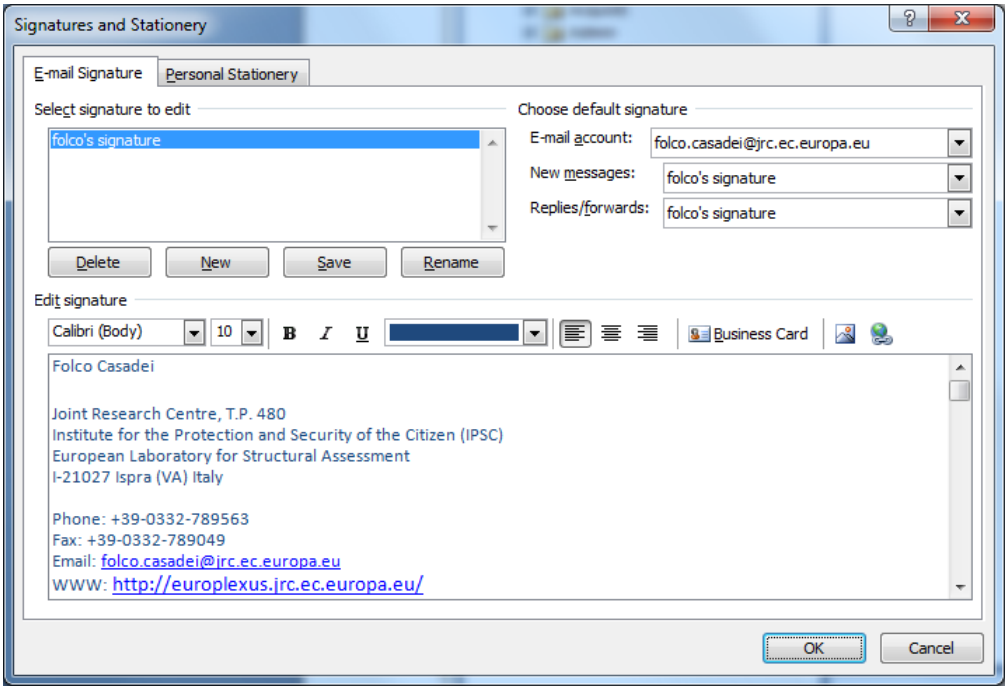
Next, click on the Mail Format tab, then on Signatures



Edit a new signature (folco's signature) as sown below, then click on OK.



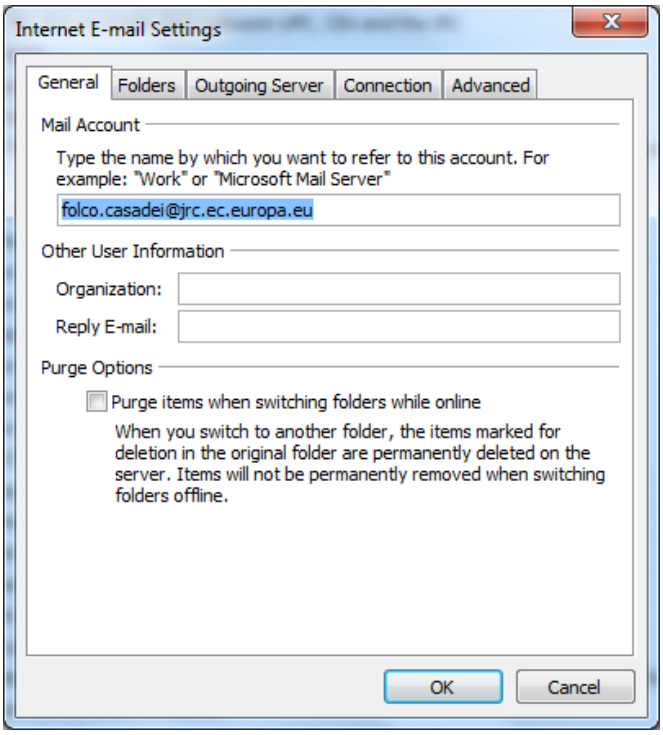
The new signature should now appear automatically in any new message created. To make sure that the signature appears automatically also in replies and in forwarded messages, make sure to choose the signature not only in the "New messages" dialog, but also in the "Replies/forwards" dialog:



10.17.17 Permanently deleting messages in MS Outlook 2007

In Outlook 2007, when one deletes a message from the Inbox, it is striked through but it remains in the mailbox (for possible later undelet). To delete it permanently, do the following:

- Click on Edit → Purge → Purge options. The following dialog appears



- Tick the "Purge items when switching folders while online" box.

To permanently delete (purge) messages in a folder, first delete dem (they get striked) and then change to another folder, then back to the original folder.

10.17.18 Finalization of the Windows 7 64-bit implementation

The following corrections have been performed to finalize the implementation of the new Windows 7 64-bit hardware:

- The compiler on \\sm12 machine has been downgraded to version 11.1.060 (from 11.1.067), so it is now the same version on all machines. This allows to simplify and generalize the compilation procedures.

- All procedures (Perl scripts) on the new \\sm47 machine (32-bit EUROPLEXUS server), formerly \\sm58, have been updated so that the new path C:\EUROPLEXUS is used instead of the old path E:\EUROPLEXUS. Sometimes this path appeared explicitly in some procedures, instead of being taken from the environment variable %EUROPLEXUS%.

- The MiKTeX (Version 2.8) has been installed on the \\sm32 machine (EUROPLEXUS 64-bit server) and the TEXINPUTS environment variable on \\sm12 now points to \\sm32 rather than to \\sm47. This simplifies the procedures for manual evolution.

- The `epx_evol_64` evolution procedure (for the \\sm32 64-bit machine) has been modified by activating the production of the manual. However, the uploading of the manuals on the on-line server remains inhibited (this is already done by the 32-bit server).

10.17.19 Correction of the epx_grep script for Windows 7 64-bit

Under Windows 7, 64 bit, the epx_grep script applied to search on the EUROPLEXUS sources fails with the error message "`Too many open files`". This did not happen under XP 64 bit, nor under Windows 7 32-bit.

The problem seems related to the too large number of source files (almost 2000). In fact, with other file types (benchmark, manual, include) the command works.

In order to avoid the problem, the `epx_grep` script is modified so that, instead of searching all files in a single execution of `grep`, it searches letter-by-letter. There are therefore 26 calls to the `grep` utility, one for each letter of the alphabet. As a consequence, only files whose names start with a letter (lowercase or uppercase, there is no difference) are inspected.

The modified command works, but it is much slower than the previous version, at least when run on \\sm12 (Windows 7 64-bit machine, using \\sm32 XP 64-bit machine as "server").

10.18 Modifications after June 2011

10.18.1 Use of Intel MKL library

Almost all installations of EUROPLEXUS on the different platformas now use the Intel Fortran compiler. This compiler comes with a library MKL wich contains, among other things, the Blas and Lapack libraries and libraries for Fast Fourier Transforms needed in EUROPLEXUS. It is therefore advised to use these built-in (and probably highly optimized) libraries instead of the ones (Libblas and Liblapack) produced by us from the corresponding sources, which were used so far.

To have access to functions in the MKL library:

- No change has to be made in the Fortran sources (no `USE` or `INCLUDE` has to be added).
- In the compilation command (`ifort`) a special switch has to be added: under Windows this is `/Qmkl:sequential` (to use the sequential, not the parallel, version of MKL). This swithch "embeds" in the resulting object files `.obj` an information for the linker (`link` command) to use the appropriate mkl library files. The version of MKL appropriate for the current platform (32 or 64-bit etc.) is set when the `ifortvars.bat` script is executed at the beginning of each `epx_cmp` or `epx_lk` command.
- No change must be done in the linking procedure (here done using the `LINK` command). The linker should then find the MKL routines in the libraries whose name has been embedded by `ifort` in the `.obj` files.

In order to conform to this new strategy, the following modifications have been implemented in the scripts:

- The `epx_cmp` compilation script is modified in such a way that the `ifort` command uses the `/Qmkl:sequential` switch by default. If you do not want to use it, there is a new optional switch `-nomkl` which disables it.
- The `epx_lk` linkage script is modified in such a way that the `LINK` command does not use the (locally made) `Libblas` and `Liblapack` libraries any more, but uses the MKL library (via information in the object files). To use the local `Libblas` and `Liblapack` libraries, add the new switch `-nomkl`.

Furthermore, the MS Visual Studio project files automatically copied by the epx_init command to allow local debugging are modified as follows:

- In the `plex` solution, right-click on the `epx` project and open the **Properties** page. In The **Fortran** tab select **Libraries** and then under **Use Intel Math Kernel Library** (which was empty) activate **Sequential (/Qmkl:sequential)**.
- In the `plex` solution, right-click on the `epx` project and open the **Properties** page. In The **Linker** tab select **Input** and then under **Additional dependencies** remove **Libblas.lib Liblapack.lib**.

10.18.2 Use of FFT functions from the Intel MKL library

The Intel MKL library contains many auxiliary functions, among which are of interest, for example, the functions for Fast Fourier Transforms (FFT). To make use of these functions, it is not sufficient to follow the procedure described in Section 10.18.1, but some additional measures should be taken.

At compilation time, it is necessary to make use of a module called `MKL_DFTI` via the f90 instruction `USE MKL_DFTI`. The corresponding `.mod` file must therefore be accessible to the comiler (which is not the case by default). In turn, this module needs a second module called `MKL_DFT_TYPE`.

The sources for both modules are found in a single `.f90` source file which, for the 11.1 version of the compiler and a 64-bit architecture, for example, is:

```
C:\Program Files (x86)\Intel\Compiler\11.1\060\mkl\include\mkl_dfti.f90
```

To compile this file, copy the file to a temporary directory, open a console window on this directory and type the commands:

```
epx_setvars (to set the necessary environment variables)

ifort /O3 /c /Qmkl:sequential mkl_dfti.f90
```

This produces the following files:

```
mkl_dfti.obj

mkl_dft_type.mod

mkl_dfti.mod
```

Copy the two `.mod` files to a location where your compiler can find them by defaults, e.g. to the same place where the OpenGL-based modules are located. On the same platform considered in the example above, this is:

```
C:\Program Files (x86)\Intel\Compiler\11.1\060\include\intel64
```

The `epx_cmp` procedure has been updated by adding a new keyword `MFFT` to the list of keywords used for the filtering. This keyword activates compilation of source code invoking the MKL FFT functions on platforms that have access to the intel MKL library.

Finally, any benchmark input file that needs to use MKL FFT functions must include the (new) command `MFFT` in the problem definition section of the input file.

10.18.3 Correction of epx_mail script

The `epx_mail` script, now based upon SendMail 2.09, seems not to work correctly: the mail message is sent to just the first of recipients and the first of cc, although no error is printed out. To overcome the problem, a new script `epx_mail_1` is created which sends the message to just one recipient at a time. Then, `epx_mail` is modified so that it invokes `epx_mail_1` repeatedly, once for each recipient.

The syntax of the `epx_mail` command is unchanged. The syntax of the `epx_mail_1` command is:

```
epx_mail_1 status num filename recipient
```

where:

`status`	`OK` or `FAILED!!!`.
`num`	Evolution number.
`filename`	Name of the evolution log file to be attached to the message.
`recipient`	Name of the recipient the message.

10.18.4 Shortening the evolution procedure

The `epx_evol_start` and the `epx_evol_64` scripts have been updated at the beginning of August 2011 in such a way that only the source files transmitted by the central mirror site are compiled, even in the presence of mudule files. Previously, if any module was present, the whole source was recompiled, but this led to unacceptably long evolution times (more than 6 hours on the \\sm47 machine). The new procedure relies upon the fact (guaranteed by the central mirror site) that not only the sources with changes, but also all the dependencies, are transmitted in the sources evolution package. Of course, from time to time, it will be a good thing to re-compile anyway the complete source in order to update the libraries in a clean way.

The `epx_evol_start and epx_evol_64` scripts now admit a new switch `-fullcmp`, which forces full source re-compilation (if need be). This may be useful for example when launching "by hand" a problematic evolution.

10.18.5 Measuring compilation times

The compilation times of some routines have become excessively long, especially when compiling with the –check option. This is due to the programming style, e.g. using too many derivations (%) in the same line with derived types (like e.g. in `a = my_obj%toto%titi%tata%tutu`). The problem may be avoided by using an intermediate pointer: `ptr => my_obj%toto%titi` and then `a = ptr%tata%tutu`, for example).

In order to find out which are the problematic routines, a new switch –times has been added to the epx_cmp script. This activates the printout in the .err file associated with each compiled .ff file of the initial and final times of the compilation. Then, by running a new utility program compil_times, one obtains a list of the compilation times of all the sources in the current directory, in decreasing order.

To monitor the compilation times on a set of routines in the current directory, use the following commands:

- `epx_cmp <optional switches> -times`: this compiles all sources (`.ff`) in the current directory with the chosen options (e.g. `-o -c`), ans produces the compilation times in the corresponding `.err` files.

- `compil_times`: this command reads all the `.err` files in the current directory, computes the compilation time for each file and then lists the results in decreasing order, so that the files which take longer to be compiled are listed first.

10.19 Mirror site status as of January 2013

This Section contains some additional information on the EUROPLEXUS mirror site implementation at JRC as of January 2013, just before the retirement of F. Casadei. This information (in addition to the contents of the previous Sections) can be useful in managing the automatic upgrade of the site during the "interim" period, until a new responsible is nominated.

10.19.1 EPX "servers"

Two EPX "servers" are currently set up at JRC:

- \\sm47 contains the 32-bit Windows implementation of the code (see \\sm47\C\Europlexus). This machine runs Windows 7 Professional (SP1), 32-bit operating system with 4 GB of RAM.

- \\sm32 contains the 64-bit Windows implementation of the code (see \\sm32\E\Europlexus). This machine runs Windows XP Professional (SP2), 64-bit operating system with 32 GB of RAM.

In addition to these two machines, F. Casadei's personal PC is available:

- \\sm12 runs Windows 7 Professional (SP1), 64-bit operating system with 8 GB of RAM.

10.19.2 Internal and external disks

The machines listed above have some external disks connected.

The machine \\sm12 has an internal disc (C:) which is configured in RAID and therefore does not require a backup. This disk has about 606 GB of used space and 324 GB of free space.

The following four external disks are connected to the \\sm12 machine:

- \\sm12\E (RAID_1) has about 811 GB of used space and 120 GB of free space. This disk contains mostly old data from previous users/developers of the code (Giannopoulos, Larcher, Kristoffersen, Rakvag etc.). This unit is configured in RAID so it is not necessary to have a backup.

- \\sm12\F (RAID_2) has about 335 GB of used space and 596 GB of free space. This disk contains mostly old data from F. Casadei. This unit is configured in RAID so it is not necessary to have a backup.

- \\sm12\G (RAID_3) has about 595 GB of used space and 335 GB of free space. This disk contains mostly old data from runs performed by F. Casadei on the 64-bit EPS server (\\sm32). This unit is configured in RAID so it is not necessary to have a backup.

- \\sm12\H (Backup) has about 833 GB of used space and 97 GB of free space. This disk is used as a backup from \\sm32 and \\sm47 (see below for details). This unit is not configured in RAID.

The machine \\sm32 has an internal disc (C:) which is not configured in RAID. This disk has about 46 GB of used space and 31 GB of free space. The machine also has an internal disk (D:), not configured in RAID, which contains about 137 GB of data (from Casadei, Bonalumi, Pegon and Valsamos), and 95 GB of free space. Finally, the machine has a third internal disk (or partition) (E:), not configured in RAID, which contains about 223 GB of data, and 10 GB of free space. The \\sm32\E\Europlexus directory occupies about 4 GB.

The following external disk is connected to the \\sm32 machine:

- \\sm32\F (GV_simulations) has about 1,420 GB of used space and 405 GB of free space. This disk contains recent calculations performed by G. Valsamos. This unit is not configured in RAID. However, the capacity of the disk is so big that no automatic backup has been set up yet for this unit!

The machine \\sm47 has an internal disc (C:) which is not configured in RAID. This disk has about 183 GB of used space and 282 GB of free space. The \\sm47\C\Europlexus directory occupies about 38 GB.

10.19.3 Backups and other scheduled processes

Some processes are scheduled on the various machines, in order to automatize some tasks such as code evolutions and backups (for the disks not configured in RAID). To check the currently scheduled tasks, use the `epx_checkat` command (see Section 10.14.10). To (re-)schedule tasks, use the `epx_schtasks` command under \\sm47, or the `epx_schtasks_64` command under \\sm32.

The following tasks are currently active on the \\sm32 machine (see `epx_schtasks_64` command on \\sm32):

- Task named `epx_evo` launches the 64-bit code evolution (command epx_evol_64) daily at 01:00. The command used is `epx_evol_64` on \\sm32:.

- Task named `bk_64_D` backs up the \\sm32\D\Users directory to \\sm12\Backup\Bk_64bit_D\Users daily at 06:30. The command used is `backup_64bit_D` on \\sm32\C:.

- Task named `bk_64_E` backs up the \\sm32\E\Cast3m directory to \\sm12\Backup\Bk_64bit_E\Cast3m daily at 07:00. Similarly, it backs up also the directories Europlexus, Data and Tests. The command used is `backup_64bit_E` on \\sm32\C:.

- Taks named `bk_epx_srv` backs up the \\sm47\C\Europlexus directory (i.e., the 32-bit version of the code!), with the exclusion of the "check" sub-directory, to \\sm12\Backup\Bk_epx_server\Europlexus, daily at 08:00. The command used is `backup_epx_server_64` on \\sm32\C:.

The following tasks are currently active on the \\sm47 machine (see `epx_schtasks` command on \\sm47):

- Task named `epx_evo` launches the 32-bit code evolution (command `epx_evol_start`) daily at 00:30. The command used is `epx_evol_start` on \\sm47.

- Task named `epx_sav` saves the entire (with a few exclusions) \\sm47\E\Europlexus directory to \\sm47\E\Europlexus\Save weekly (every sunday) at 12:00. The command used is `epx_save` on \\sm47.

- Task named `epx_chk` launches the "-check" evolution of the code on \\sm47 daily at 18:30. The command used is `epx_evol_check` on \\sm47.

No tasks are currently scheduled on the \\sm12 machine.

10.20 Setup with Visual Studio 2010 and Intel Fortran 2013

In January 2013 an installation has been made on a DELL Precision M6700 laptop (DELL_M6700) with 32 GB of ram, 8-core processor (i7-3720QM), running Windows 7 Professional SP1, 64-bit. The following basic software has been installed on this machine:

- MS Visual Studio 2010 version 10.0.40219.1 SP1rel.

- Intel 2013 Fortran compiler (Intel Parallel Studio XE 2013 with VS2010), version 13.0.1.119 Build 20121008.

It was the first time that an implementation under this environment was attempted.

10.20.1 EPX implementation

After installing Perl, Ghostscript/Ghostviwe, Netpbm and Vim as usual (see Section 10.17.15 for vim's customization), the Europlexus software was installed as follows.

- EPX_2376 (last version of 2012) was copied from \\sm47 via epx_save. Only the sources are retained, since \\sm47 is a 32-bit machine. The libraries, procedures etc. are then taken from \\sm32 (64-bit machine).

- The directories Run, Run\Gnu, Proc, Trace, Manual_filtered, Init, Util and Library are copied from \\sm32.

10.20.2 Epx_setvars

The `epx_setvars.bat` command has to be customized because the new compiler is installed on different directories. The line:

```
SET IFV=C:\Program Files (x86)\Intel\Compiler\11.1\060\Bin\ifortvars.bat
```

becomes

```
SET IFV=C:\Program Files (x86)\Intel\Composer XE\Bin\ifortvars.bat
```

Note that in the new machine there are two directories: "Composer XE" and "Composer XE 2013". It seems that the first one is a link to the second, probably to use a fixed name, that will not change as new releases of the copiler will appear. For this reason, the generic name is used in the epx_setvars procedure.

The `ifortvars.bat` procedure is much clearer than on the previous release, and now it seems that it should be possible, with little effort, to produce both 64-bit and 32-bit versions of the code on the same (64-bit) machine. However, this is left for further investigation.

From the Start menu it is possible to open a command window configured either for 64-bit, or for 32-bit compilation:

- The 64-bit command window prints "Setting environment for using Microsoft Visual Studio 2010 x64 tools" (this phrase appears in the script `C:\Program Files (x86)\Intel\Composer XE 2013\Bin\vsshell2010vars_arch.bat`.
- The 32-bit command window prints "Setting environment for using Microsoft Visual Studio 2010 x86 tools".

10.20.3 Libraries

The Glut-related includes `glut.h`, `glutf90.h` and `glaux.h` are copied from:

```
\\sm12\C\Program Files\Microsoft SDKs\Windows\V6.0A\include\gl
```

to

```
DELL M6700\C\Program Files (x86)\Microsoft SDKs\Windows\V7.0A\include\gl
```

The `glut32.lib` library is copied from:

```
\\sm12\C\Program Files\Microsoft SDKs\Windows\V6.0A\lib\x64
```

to

```
DELL M6700\C\Program Files (x86)\Microsoft SDKs\Windows\V7.0A\lib\x64
```

The libraries `bmplib.lib`, `f90gl.lib`, `f90glu.lib`, `f90glut.lib`, are copied from:

```
\\sm12\C\Program Files (x86)\Intel\Compiler\11.1\060\lib\intel64
```

to

```
DELL M6700\C\Program Files (x86)\Intel\Composer XE 2013\Compiler\lib\intel64
```

The mudule files `opengl_fwrap.mod`, `opengl_gl.mod`, `opengl_glinterfaces.mod`, `opengl_glu.mod`, `opengl_gluinterfaces.mod`, `opengl_glut.mod`, `opengl_glutinterfaces.mod`, `opengl_kinds.mod`, `mkl_dft_type.mod` and `mkl_dfti.mod` are copied from:

```
\\sm12\C\Program Files (x86)\Intel\Compiler\11.1\060\include\intel64
```

to

```
DELL M6700\C\Program Files (x86)\Intel\Composer XE 2013\Compiler\include\intel64
```

After these settings, the complete compilation (`epx_cmp -o`) and link (`epx_lk -o`) work correctly from the command line. Complete optimized compilation takes about 40 minutes. All benchmarks are passed without problems in 37 minutes (70 minutes on \\sm47 and 90 minutes on \\sm32).

10.20.4 Epx_init

Next, the debuggable version is set up under MS Visual Studio 2010. The "solution" is copied from \\sm32 and opened on the new machine: the conversion is OK with only some warnings. Compilation of the new solution runs smoothly, but at link time the following error appers:

```
   Error 1 LNK2005: _initp_misc_inivarg already defined in LIBCMTD.lib (inivarg.obj)
[file LIBCMT.lib (inivarf.obj)]
```

A total of 10 such messages appears. Furthermore, the following warning appears:

```
   Warning LNK4098: defaultlib 'LIBCMT' conflicts with use of other libs, use
/NODEFAULTLIB
```

To avoid this problem, the `LIBCMT.lib` library is tentatively excluded from the project. In the `plex` solution, right-click on the `epx` project, Properties, Linker, Input, Ignore Specific Library, type `libcmt.lib` in the dialog box. Now the solution compiles and links correctly.

The `epx_init` script has to be modified because the new compiler/visual environment uses additional files. From the converted solution (2008 to 2010) we see that these files are the following:

- `plex.sdf`
- `filter\filter.vcxproj`
- `filter\filter.vcxproj.filters`
- `filter\filter.vcxproj.user`
- `ff_rule.props`
- `ff_rule.targets`
- `ff_rule.xml`

With these additions the `epx_init` command works correctly, and prepares a debuggableversion of the code. The source files can then be added to the solution as usual.

10.20.5 MiKTeX

The MiKTeX package version 2.8 is installed as described in Section 10.16.4, including also Tth and Hevea. The compilation of the User's manual (all versions) proceeds smoothly.

11 References

1. A. Hoffmann, M. Lepareux, Ph. Jamet, B. Schwab, A. Forestier, H. Bung: "PLEXUS. A General Computer Program for Fast Dynamic Analysis", Rapport DEMT 86/295, 1986.

2. H. Bung, F. Casadei, J.P. Halleux, M. Lepareux: "PLEXIS-3C: a Computer Code for Fast Dynamic Problems in Structures and Fluids", 10th International Conference in Structural Mechanics in Reactor Technology, SMiRT-10, Anaheim, USA, August 14-19, 1989.

3. H. Bung, F. Casadei, J.P. Halleux, M. Lepareux: "Specifications for the Multi-Site Development of the EUROPLEXUS Computer Code", JRC Technical Note N. I.00.147, December 2000.

4. H. Bung, M. Lepareux: "PLEXUS: Atelier de Logiciels KSTM", CEA internal document, 1999.

5. F. Casadei, J.P. Halleux, H. Bung, M. Lepareux: "Some Tentative Guidelines for the Development of the EUROPLEXUS Software System", JRC Technical Note N. I.00.146, December 2000.

6. F. Casadei: “Implementation of the PLEXIS-3C System Under MS-Windows NT”, Technical Note N. I.98.78, May 1998.

7. L. Lamport: “LaTeX: A Document Preparation System. User’s Guide and Reference Manual”, Addison-Wesley, 1986.

8. D.E. Knuth: “The TeXbook”, Addison-Wesley, Reading, Massachusetts, 1984.

9. L. Lamport: “LaTeX: A Document Preparation System. User’s Guide and Reference Manual. (Updated for LaTeX2e)”, Addison-Wesley, 1994.

10. F. Casadei, J.P. Halleux, H. Bung, M. Lepareux: “Organisation of the EUROPLEXUS Mirror Site (Windows NT) at JRC Ispra”, JRC Technical Note N. I.00.145, December 2000.

11. F. Casadei, J.P. Halleux, H. Bung, M. Lepareux: “Organisation of the EUROPLEXUS Mirror Site (MS Windows) at JRC Ispra – Second Edition”, JRC Technical Note N. I.02.03, January 2002.

12. F. Casadei: “Adaptation of OpenGL Libraries for Use Within the EUROPLEXUS System”, JRC Technical Note N. I.03.26, February 2003.

13. F. Casadei: “Generation of Bitmapped OpenGL Graphical Images Within EUROPLEXUS”, JRC Technical Note N. I.03.68, April 2003.

14. Ph. Buchet, F. Casadei, P. Pegon: “A Module for Advanced 3D OpenGL Graphics Rendering in Computational Mechanics”, JRC Technical Note N. I.03.169, November 2003.

15. F. Casadei, J.P. Halleux, H. Bung, M. Lepareux: “Organisation of the EUROPLEXUS Mirror Site (MS Windows) at JRC Ispra – Third Edition”, JRC Technical Note N. I.03.70, April 2003.

16. F. Casadei, J.P. Halleux, H. Bung, M. Lepareux: “Organisation of the EUROPLEXUS Mirror Site (MS Windows) at JRC Ispra – Fourth Edition”, JRC Publication N. S.P.I.05.166.EN, October 2005.

17. F. Casadei, H. Bung: “Organisation of the EUROPLEXUS Mirror Site (MS Windows) at JRC Ispra – Fifth Edition”, JRC Technical Note PUBSY No. JRC40467, October 2007.

18. F. Casadei, H. Bung: “Organisation of the EUROPLEXUS Mirror Site (MS Windows) at JRC Ispra – Sixth Edition”, JRC Technical Note PUBSY No. JRC48621, November 2008.

12 Appendix

This Section contains the listings of all the files mentioned or described in the preceding Sections.

autodecl.f

```
      program autodecl
c
      USE dflib ! For GETARG, NARGS
c
c Automatically produce FORTRAN declarations of undefined variables
c (for use with IMPLICIT NONE) starting from compilation error file
c
c IMPORTANT: it is assumed that the usual IMPLICIT REAL*8 (A-H,O-Z)
c statement is used in the source, i.e. all I,J,K,L,M,N variables are
c integer, the rest are real*8 !!!
c
      implicit none
c
      character line*255
      integer, parameter :: mxvar=2000
      character*10 var,vari(mxvar), vard(mxvar)
      integer ni, nd
c
      character err1*39, err2*30, err*69, ini*1, name*12, inp*12,
     &          out*12
      integer leng, kk, i1, i2
c
      integer nar
      INTEGER(2) :: pos, stat
c
      data err1/'Error: This name does not have a type, '/,
     *     err2/'and must have an explicit type'/
c
      err=err1//err2
      ni = 0
      nd = 0
c
c Check number of arguments
c
      nar = NARGS() ! Number of command line arguments, including the command
      IF (nar /= 2) THEN
        write(0,'(a)') 'WRONG NUMBER OF COMMAND LINE ARGUMENTS'
        STOP
      ENDIF
c
c Retrieve command line name
c
      pos = 1
      CALL GETARG (pos, name, stat)
      IF (stat <= 0) THEN
        write (0,'(a)') 'Wrong command line argument'
        STOP
      ENDIF
c
      inp = TRIM(name)//'.err'
      out = TRIM(name)//'.dcl'
      open(unit=1,file=inp,form='FORMATTED',err=991)
      open(unit=2,file=out,form='FORMATTED',err=992)
c
    1 read(1,'(a)',end=99) line
      if( INDEX(line, err) /= 0) then
        i1 = INDEX (line, '[', .TRUE.)
        i2 = INDEX (line, ']', .TRUE.)
        if (i1 /= 0 .AND. i2 /= 0) then
          i1 = i1 + 1
          i2 = i2 - 1
          leng = i2 - i1 + 1
          if (leng > 10) then
            write(0,*) 'Var name too long:', leng
            stop
          endif
          var = line(i1:i2)
          ini = var(1:1)
          if ((LGE(ini, 'i') .AND. LLE(ini, 'n')) .OR.
     &        (LGE(ini, 'I') .AND. LLE(ini, 'N'))    ) then
            ni = ni + 1
            vari(ni) = var
          else
            nd = nd + 1
            vard(nd) = var
          endif
        endif
      endif
      go to 1
c
   99 if (ni > 0) then
        do kk=1,ni-1
        vari(kk)=TRIM(vari(kk))//','
        end do
        write(2,*) '     integer'
        write(2,2) (vari(kk),kk=1,ni)
    2   format('     & ',a11,a11,a11,a11,a11)
      endif
      if (nd > 0) then
        do kk=1,nd-1
        vard(kk)=TRIM(vard(kk))//','
        end do
        write(2,*) '     real*8'
        write(2,3) (vard(kk),kk=1,nd)
    3   format('     & ',a11,a11,a11,a11,a11)
      endif
      stop
c
  991 write(0,'(a,a)') 'ERROR while opening input file:',inp
      stop
  992 write(0,'(a,a)') 'ERROR while opening output file:',out
      stop
c
      end
```

autodecl_dum.f

```
      program autodecl
c
      USE dflib ! For GETARG, NARGS
c
c Automatically produce FORTRAN declarations of undefined variables
c (for use with IMPLICIT NONE) starting from compilation error file
c
c IMPORTANT: it is assumed that the usual IMPLICIT REAL*8 (A-H,O-Z)
c statement is used in the source, i.e. all I,J,K,L,M,N variables are
c integer, the rest are real*8 !!!
c
      implicit none
c
      character line*255
      integer, parameter :: mxvar=2000
      character*10 var,vari(mxvar), vard(mxvar)
      integer ni, nd
c
      character err1*36, err2*17, err*53, ini*1, name*12, inp*12,
     &          out*12
      integer leng, kk, i1, i2
c
      integer nar
      INTEGER(2) :: pos, stat
c
      data err1/'Error: This name has not been given '/,
     *     err2/'an explicit type.'/
c
      err=err1//err2
      ni = 0
      nd = 0
c
c Check number of arguments
c
      nar = NARGS() ! Number of command line arguments, including the command
      IF (nar /= 2) THEN
        write(0,'(a)') 'WRONG NUMBER OF COMMAND LINE ARGUMENTS'
        STOP
      ENDIF
c
c Retrieve command line name
c
      pos = 1
      CALL GETARG (pos, name, stat)
      IF (stat <= 0) THEN
        write (0,'(a)') 'Wrong command line argument'
        STOP
      ENDIF
c
      inp = TRIM(name)//'.err'
      out = TRIM(name)//'.dcl'
      open(unit=1,file=inp,form='FORMATTED',err=991)
      open(unit=2,file=out,form='FORMATTED',err=992)
c
    1 read(1,'(a)',end=99) line
      if( INDEX(line, err) /= 0) then
        i1 = INDEX (line, '[', .TRUE.)
        i2 = INDEX (line, ']', .TRUE.)
        if (i1 /= 0 .AND. i2 /= 0) then
          i1 = i1 + 1
          i2 = i2 - 1
          leng = i2 - i1 + 1
          if (leng > 10) then
            write(0,*) 'Var name too long:', leng
            stop
          endif
          var = line(i1:i2)
          ini = var(1:1)
          if ((LGE(ini, 'i') .AND. LLE(ini, 'n')) .OR.
     &        (LGE(ini, 'I') .AND. LLE(ini, 'N'))    ) then
            ni = ni + 1
            vari(ni) = var
          else
            nd = nd + 1
            vard(nd) = var
          endif
        endif
      endif
      go to 1
c
   99 if (ni > 0) then
        do kk=1,ni-1
        vari(kk)=TRIM(vari(kk))//','
        end do
        write(2,*) '     integer'
        write(2,2) (vari(kk),kk=1,ni)
    2   format('     & ',a11,a11,a11,a11,a11)
      endif
      if (nd > 0) then
        do kk=1,nd-1
        vard(kk)=TRIM(vard(kk))//','
        end do
        write(2,*) '     real*8'
        write(2,3) (vard(kk),kk=1,nd)
    3   format('     & ',a11,a11,a11,a11,a11)
      endif
      stop
c
  991 write(0,'(a,a)') 'ERROR while opening input file:',inp
      stop
  992 write(0,'(a,a)') 'ERROR while opening output file:',out
      stop
c
      end
```

autoqual.f

```
      PROGRAM AUTOQUAL
*
* Read EUROPLEXUS listing and input files, searching for qualifications
* (the two files must match each other, i.e. the listing must be
*  the result of running EUROPLEXUS with the given input file)
*
* Modify the input file so that the qualification values as taken
* from the listing are exactly satisfied.
*
      USE dflib ! for GETARG, NARGS
*
      IMPLICIT NONE
*
      CHARACTER*32 base,  ! base name of input and listing files
```

```
     >             ifile, ! input file name
     >             jfile, ! modified input file name
     >             lfile  ! listing file name
      INTEGER, PARAMETER :: nquamax = 50 ! max no. of qualifications
      CHARACTER*4 :: quantity(nquamax)
      CHARACTER*12 :: expected(nquamax), found(nquamax),
     >                tolerance(nquamax), relerr(nquamax)
      CHARACTER*4 :: qu
      CHARACTER*12 :: ex, fo, tol, re, cfactol
      REAL*8 :: r8ex, r8fo, factol, tolnew, tolold
      INTEGER :: nqua       ! number of qualifications read in
      INTEGER :: nquaw      ! number of qualifications written
      CHARACTER*161 :: line, ! line read in from the listing or input files
     >                 uline,! same but uppercase
     >                 aline ! same but uppercase and left-adjusted
      INTEGER :: nar, pos, stat, eof, iad_refe, iad_tole, len_refe
*
      nqua = 0
      nquaw = 0
      factol = 1.D0 ! multiplicative factor for tolerances
*
* Check number of arguments
      nar = NARGS() ! Number of command line arguments, including the command
      IF (nar < 2) STOP 'WRONG NUMBER OF COMMAND LINE ARGUMENTS'
*
* Retrieve base name from command line argument 1
      pos = 1
      CALL GETARG (pos, base, stat)
      IF (stat <= 0) STOP 'Wrong command line argument: base name'
*
* Retrieve factol from command line argument 2 (if present)
      IF (nar > 2) THEN
        pos = 2
        CALL GETARG (pos, cfactol, stat)
        IF (stat <= 0) STOP 'Wrong command line argument: factol'
        READ (cfactol, *) factol
        IF (factol <= 0.D0) STOP 'factol <= 0.D0'
      ENDIF
*
      lfile = TRIM(base) // '.listing'
      ifile = TRIM(base) // '.epx'
      jfile = TRIM(base) // '.new'
*
      OPEN (1, FILE=lfile, ERR=901)
*
* Search for 'QUALIFICATIONS' in listing
*
    1 CALL read_qual (qu, ex, fo, tol, re, eof)
      IF (eof == 1) GO TO 2
      nqua = nqua + 1
      IF (nqua > nquamax) STOP 'NQUA > NQUAMAX'
      quantity(nqua) = qu
      expected(nqua) = ex
*
* set to 0 very small nonzero found value, if expected value was exactly 0
      READ (ex, *) r8ex
      READ (fo, *) r8fo
      IF (r8ex == 0.D0 .AND. ABS(r8fo) < 1.E-10) fo =' 0.00000D+00'
*
      found(nqua) = fo
      tolerance(nqua) = tol
      relerr(nqua) = re
      GO TO 1
*
    2 IF (nqua == 0) STOP 'NQUA = 0'
      OPEN (2, FILE=ifile, ERR=902)
      OPEN (3, FILE=jfile, ERR=903)
*
    3 READ (2, 1001, END=900) line
      uline = line
      CALL to_upper (uline)
      aline = ADJUSTL (uline)
      IF (aline(1:5) /= 'QUAL ') THEN
        WRITE (3, 1001) TRIM (line) ! copy line as-is
        GO TO 3
      ENDIF
*
* input line containing "QUAL " was found.
* Scan this line and the following ones:
*   - if it is a comment line (starts by C or by *) copy it as-is
*   - else, if both "REFE" and "TOLE" are found on the line, it is considered
*     a qualifiation line and is edited, then copied
*   - else, it is a non-qualification line, copy it as such and
*     then continue reading until a new "QUAL " is (possibly) found
*
      aline = ADJUSTL (aline(5:)) ! remove initial "QUAL"
*
    4 IF (uline(1:1) == 'C' .OR. uline(1:1) == '*') GO TO 5 ! comment line
      iad_refe = INDEX (aline, 'REFE ')
      IF (iad_refe == 0) THEN
        WRITE (3, 1001) TRIM (line)
        GO TO 3
      ENDIF
      iad_tole = INDEX (aline, 'TOLE ')
      IF (iad_tole == 0) THEN
        WRITE (3, 1001) TRIM (line)
        GO TO 3
      ENDIF
      nquaw = nquaw + 1
      IF (quantity(nquaw) /= aline(1:4))
     >  STOP 'quantity(nquaw) /= aline(1:4)'
*
* replace old reference value by value found in the listing
      iad_refe = INDEX (uline, 'REFE ')
      IF (iad_refe == 0) STOP 'iad_refe == 0'
      iad_tole = INDEX (uline, 'TOLE ')
      len_refe = iad_tole - iad_refe
      IF (len_refe < 3) STOP 'len_refe < 3'
      IF (len_refe < 18) THEN ! make refe field exactly 14 chars wide
        line = line(1:iad_refe+3) // '              ' //
     >         line(iad_tole:)
        uline = line
        CALL to_upper (uline)
      ENDIF
      iad_tole = INDEX (uline, 'TOLE ')
      line(iad_refe+4:iad_tole-1) = ' ' // found(nquaw)
*
* if factol /= 1.D0, multiply old tolerance by factol
      IF (factol /= 1.D0) THEN
        READ (tolerance(nquaw), *) tolold
        tolnew = factol*tolold
        WRITE (line(iad_tole+4:), '(1PE12.5)') tolnew
      ENDIF
*
      IF (LEN_TRIM (line) > 72) STOP 'Edited line > 72 characters!'
    5 WRITE (3, 1001) TRIM (line) ! copy modified line
      READ (2, 1001, END=900) line
      uline = line
      CALL to_upper (uline)
      aline = ADJUSTL (uline)
      GO TO 4
*
  900 STOP 'Normal end'
*
  901 STOP 'Error opening listing file.'
  902 STOP 'Error opening input file.'
  903 STOP 'Error opening modified input file.'
*
 1001 FORMAT (A)
*
      END PROGRAM AUTOQUAL
*=======================================================================
      SUBROUTINE READ_QUAL (quantity, expected, found, tolerance,
     >                      relerr, eof)
*
      IMPLICIT NONE
*
* args
      CHARACTER*4, INTENT(OUT) :: quantity
      CHARACTER*12, INTENT(OUT) :: expected, found, tolerance, relerr
      INTEGER, INTENT(OUT) :: eof
*
* locals
      CHARACTER*161 :: line ! line read in from the listing or input files
*
      eof = 0
*
    1 READ (1, 1001, END=900) line
      line = ADJUSTL (line)
      IF (line(1:18) == 'MONITORED QUANTITY') THEN
        quantity = line(37:40)
*
        READ (1, 1001, END=900) line
        line = ADJUSTL (line)
        IF (line(1:9) /= 'COMPONENT')
     >    STOP 'COMPONENT not found'
*
        READ (1, 1001, END=900) line
        line = ADJUSTL (line)
        IF (line(1:15) /= 'NODE OR ELEMENT')
     >    STOP 'NODE OR ELEMENT not found'
*
        READ (1, 1001, END=900) line
        line = ADJUSTL (line)
        IF (line(1:14) /= 'EXPECTED VALUE')
     >    STOP 'EXPECTED VALUE not found'
        expected = line(37:48)
*
        READ (1, 1001, END=900) line
        line = ADJUSTL (line)
        IF (line(1:11) /= 'FOUND VALUE')
     >    STOP 'FOUND VALUE not found'
        found = line(37:48)
*
        READ (1, 1001, END=900) line
        line = ADJUSTL (line)
        IF (line(1:9) /= 'TOLERANCE')
     >    STOP 'TOLERANCE not found'
        tolerance = line(37:48)
*
        READ (1, 1001, END=900) line
        line = ADJUSTL (line)
        IF (line(1:14) /= 'RELATIVE ERROR')
     >    STOP 'RELATIVE ERROR not found'
        relerr = line(37:48)
*
        RETURN
      ENDIF
      GO TO 1
*
  900 eof = 1
      RETURN
*
 1001 FORMAT (A)
*
      END SUBROUTINE READ_QUAL
*=======================================================================
      SUBROUTINE to_upper (s)
c
      IMPLICIT NONE
c
      CHARACTER (LEN=*), INTENT(INOUT) :: s
c
      INTEGER(4) :: i, k
c
      DO i = 1, LEN_TRIM (s)
        k = ICHAR (s (i:i))
        IF (k >= 97 .and. k <= 122) THEN
          s (i:i) = CHAR (k - 32)
        ENDIF
      END DO
c
      END SUBROUTINE to_upper
```

compil_times.f

```
      PROGRAM compil_times
*
* Read all <base>.err files in current directory (assuming they are
* resulting from compilation of <base>.ff) and search for:
* Starting time: 1234567890
* Ending time  : 1234567890
* (where 1234567890 is machine time, in seconds sine epoch)
* Then by difference we have the compilation time in seconds.
```

```
* Order results in decreasing order and print report
*
      USE IFPORT ! for system
*
      IMPLICIT NONE
*
      INTEGER :: i, errnum, err, ir
      CHARACTER :: fname*32, line*132
      INTEGER :: nfiles, cputot
      CHARACTER*32, POINTER :: fnames(:)
      INTEGER, POINTER :: cpu(:), iran(:)
      INTEGER*8 :: i1, i2
*
      i = SYSTEM ("ls *.err > _errfiles.txt")
      IF (i == -1) THEN
        errnum = IERRNO ( )
        PRINT *, 'Error ', errnum
      ENDIF
*
      OPEN (UNIT=11, FILE="_errfiles.txt", FORM='FORMATTED',
     >      STATUS='OLD', ERR=901)
*
      nfiles = 0
    1 READ (11, 1001, END=10) fname
      nfiles = nfiles + 1
      GO TO 1
*
   10 IF (nfiles == 0) STOP 'No err files were found'
      REWIND (11)
      ALLOCATE (fnames(1:nfiles), STAT=err)
      IF (err /= 0) STOP 'Cant allocate fnames'
      ALLOCATE (cpu(1:nfiles), STAT=err)
      IF (err /= 0) STOP 'Cant allocate cpu'
      ALLOCATE (iran(1:nfiles), STAT=err)
      IF (err /= 0) STOP 'Cant allocate iran'
*
      nfiles = 0
   11 READ (11, 1001, END=20) fname
        nfiles = nfiles + 1
        fnames(nfiles) = fname
      GO TO 11
*
   20 CLOSE (11)
*
      cputot = 0
      DO i = 1, nfiles
        fname = fnames(i)
        OPEN (UNIT=11, FILE=fname, FORM='FORMATTED',
     >        STATUS='OLD', ERR=902)
        i1 = 0
        i2 = 0
   21   READ (11, 1001, END=22) line
        IF (line(1:14) == "Starting time:") THEN
          IF (i2 /= 0) STOP 'Starting time found after ending time'
          READ (line(15:), *) i1
        ELSEIF (line(1:14) == "Ending time  :") THEN
          IF (i1 == 0) STOP 'Ending time found before ending time'
          READ (line(15:), *) i2
          GO TO 22
        ENDIF
        GO TO 21
   22   CLOSE (11)
        IF (i1 == 0) PRINT *, 'Missing starting time in file', fname
        IF (i2 == 0) PRINT *, 'Missing ending time in file', fname
        IF (i1 > 0 .AND. i2 > 0) THEN
          cpu(i) = i2 - i1
        ELSE
          cpu(i) = 0
        ENDIF
        cputot = cputot + cpu(i)
*       PRINT *, 'File: ', fname, ' CPU:', cpu(i), ' s'
      END DO
*
      PRINT *, 'Total compilation time:', cputot, ' s'
*
* list files in decreasing cpu time order
*
      CALL ICLASS (nfiles, cpu, iran)
      DO ir = nfiles, 1, -1
        i = iran(ir)
        PRINT *, 'File: ', fnames(i), ' CPU:', cpu(i), ' s'
      END DO
*
      DEALLOCATE (fnames)
      DEALLOCATE (cpu)
      DEALLOCATE (iran)
      STOP 'Normal end'
*
  901 STOP 'Cant open file _errfiles.txt'
  902 PRINT *, 'Cant open file ', fname
*
 1001 FORMAT (A32)
*
      END PROGRAM compil_times

*=====================================================================

      SUBROUTINE ICLASS(N,ITAB,IRAN)
*
*     ------------------------------------------------------------------
*
*         classement des el. de itab (rang ds iran)
*         suivant la configuration, nous allons brancher
*         soit sur l'algorithme de hoare-singleton ou
*         le tri par casier (bucket)
*                                                       h.   bung  03-06
*     ------------------------------------------------------------------
*
*             n     : dimension de itab et iran
*             itab  : tableau a classer (non modifie)
*             iran  : classement
*                     l'element de "itab" classe ieme est itab(iran(i))
*             si n .le.1 ===> iran(1)=1
*
*
      IMPLICIT NONE
*
*
      INTEGER, INTENT(IN)  :: ITAB(*),N
      INTEGER, INTENT(OUT) :: IRAN(*)

      INTEGER :: IMIN,IMAX,LON,IAUX

      IF( N < 2) THEN
         IRAN(1) = 1
       RETURN
      ENDIF

      IMIN=MINVAL(ITAB(1:N))
      IMAX=MAXVAL(ITAB(1:N))
      LON = IMAX - IMIN + 1
      IAUX = N * LOG(REAL(N))*1.44D0 ! = N*LOG2(N)

      IF ((LON+N)< IAUX) THEN
!---           ON UTILISE L'ALGORITHME DE TRI PAR CASIER
         CALL ICLASS_BUCKET(N,ITAB,IRAN)
      ELSE
!---            on utilise le tri  HOARE-SINGLETON
         CALL ICLASS_HSJWM(N,ITAB,IRAN)
      ENDIF

      END SUBROUTINE ICLASS

*=====================================================================
*
      SUBROUTINE ICLASS_BUCKET(N,ITAB,IRAN)
*
*     ------------------------------------------------------------------
*
*     tri rapide utilisant le tri casier ordre croissant
*
*                                                 p.  galon 03-2006
*     ------------------------------------------------------------------
*
*             n        : longueur de itab
*             itab     : tableau non modifié
*             iran     : classement
*                        l'element de "itab" classe ieme est :          ))
*                        itab(iran(i))
*             si n .le.1 ===> iran(1)=1
*
*
* remarque : cet algorithme est performant pour n grand, disons > 100000
* et lorsque le nombre d'entiers entre le min et le max de itab est
* inferieur à nln n. la complexiteé de l'algorithme est toujours de l'or
* de o(2n+m) ou m est le cardinal de ntrav : (max -min + 1)
*
      IMPLICIT NONE
*
*
      INTEGER, INTENT(OUT) :: IRAN(*)
      INTEGER, INTENT(IN)    :: N,ITAB(*)
*
      INTEGER :: I, LONG, MIN, MAX, J, K
      INTEGER , ALLOCATABLE :: NTRAV(:)
*
*
      IRAN(1) = 1
      IF(N < 2) RETURN
*
* --- valeurs mini et maxi du tableau et longueur
*
      MIN = MINVAL(ITAB(1:N))
      MAX = MAXVAL(ITAB(1:N))
      LONG = MAX - MIN + 1
*
      ALLOCATE (NTRAV(LONG))
*
* --- initialisation du tableau
*
      NTRAV(:) = 0
*
* --- on compte les occurences de chaque entier
*
      DO I = 1, N
         K = ITAB(I)
         J = 1 + K - MIN
         NTRAV(J) = NTRAV(J) + 1
      END DO
*
* --- on calcule la somme des occurences
*
      DO I=2, LONG
         NTRAV(I) = NTRAV(I) + NTRAV(I-1)
      END DO
*
! --- On construit le tableau IRAN
*
      DO I = 1, N
         K= ITAB(I)
            J = 1 + K - MIN
       IRAN(NTRAV(J)) = I
         NTRAV(J) = NTRAV(J) - 1
      END DO
*
      DEALLOCATE (NTRAV)

*
      END SUBROUTINE ICLASS_BUCKET

*=====================================================================

      SUBROUTINE ICLASS_HSJWM(N,IX,IND)

!!    Tri Hoare-Singleton, Fortran 77 ecrits par Jones & WISNIEWSKI,
!!                         Fortran90  Alan Miller (QSORTD)

! Code converted using TO_F90 by Alan Miller
! Date: 2002-12-18  Time: 11:55:47

      IMPLICIT NONE

      INTEGER, INTENT(IN)    :: N
      INTEGER, INTENT(IN)    :: IX(*)
      INTEGER, INTENT(OUT)   :: IND(*)
```

```
!      ****************************************************************

!                                                    ROBERT RENKA
!                                            OAK RIDGE NATL. LAB.

!   THIS SUBROUTINE USES AN ORDER N*LOG(N) QUICK SORT TO SORT A REAL (8)
! ARRAY X INTO INCREASING ORDER.  THE ALGORITHM IS AS FOLLOWS.  IND IS
! INITIALIZED TO THE ORDERED SEQUENCE OF INDICES 1,...,N, AND ALL INTERCHANGES
! ARE APPLIED TO IND.  X IS DIVIDED INTO TWO PORTIONS BY PICKING A CENTRAL
! ELEMENT T.  THE FIRST AND LAST ELEMENTS ARE COMPARED WITH T, AND
! INTERCHANGES ARE APPLIED AS NECESSARY SO THAT THE THREE VALUES ARE IN
! ASCENDING ORDER.  INTERCHANGES ARE THEN APPLIED SO THAT ALL ELEMENTS
! GREATER THAN T ARE IN THE UPPER PORTION OF THE ARRAY AND ALL ELEMENTS
! LESS THAN T ARE IN THE LOWER PORTION.  THE UPPER AND LOWER INDICES OF ONE
! OF THE PORTIONS ARE SAVED IN LOCAL ARRAYS, AND THE PROCESS IS REPEATED
! ITERATIVELY ON THE OTHER PORTION.  WHEN A PORTION IS COMPLETELY SORTED,
! THE PROCESS BEGINS AGAIN BY RETRIEVING THE INDICES BOUNDING ANOTHER
! UNSORTED PORTION.

! INPUT PARAMETERS -   N - LENGTH OF THE ARRAY X.

!                     IX - VECTOR OF LENGTH N TO BE SORTED.

!                    IND - VECTOR OF LENGTH >= N.

! N AND X ARE NOT ALTERED BY THIS ROUTINE.

! OUTPUT PARAMETER - IND - SEQUENCE OF INDICES 1,...,N PERMUTED IN THE SAME
!                          FASHION AS X WOULD BE.  THUS, THE ORDERING ON
!                          X IS DEFINED BY Y(I) = X(IND(I)).

!*********************************************************************

! NOTE -- IU AND IL MUST BE DIMENSIONED >= LOG(N) WHERE LOG HAS BASE 2.

!*********************************************************************

      INTEGER   :: IU(33), IL(33)
      INTEGER   :: M, I, J, K, L, IJ, IT, ITT, INDX
      REAL      :: R
      INTEGER   :: KT

! LOCAL PARAMETERS -

! IU,IL =  TEMPORARY STORAGE FOR THE UPPER AND LOWER
!            INDICES OF PORTIONS OF THE ARRAY X
! M =      INDEX FOR IU AND IL
! I,J =    LOWER AND UPPER INDICES OF A PORTION OF X
! K,L =    INDICES IN THE RANGE I,...,J
! IJ =     RANDOMLY CHOSEN INDEX BETWEEN I AND J
! IT,ITT = TEMPORARY STORAGE FOR INTERCHANGES IN IND
! INDX =   TEMPORARY INDEX FOR X
! R =      PSEUDO RANDOM NUMBER FOR GENERATING IJ
! T =      CENTRAL ELEMENT OF X

      IF (N <= 0) RETURN

! INITIALIZE IND, M, I, J, AND R

      DO  I = 1, N
        IND(I) = I
      END DO
      M = 1
      I = 1
      J = N
      R = .375

! TOP OF LOOP

20    IF (I >= J) GO TO 70
      IF (R <= .5898437) THEN
        R = R + .0390625
      ELSE
        R = R - .21875
      END IF

! INITIALIZE K

30    K = I

! SELECT A CENTRAL ELEMENT OF X AND SAVE IT IN T

      IJ = I + R*(J-I)
      IT = IND(IJ)
      KT = IX(IT)

! IF THE FIRST ELEMENT OF THE ARRAY IS GREATER THAN kT,
!   INTERCHANGE IT WITH kT

      INDX = IND(I)
      IF (IX(INDX) > KT) THEN
        IND(IJ) = INDX
        IND(I) = IT
        IT = INDX
        KT = IX(IT)
      END IF

! INITIALIZE L

      L = J

! IF THE LAST ELEMENT OF THE ARRAY IS LESS THAN T,
!   INTERCHANGE IT WITH T

      INDX = IND(J)
      IF (IX(INDX) >= KT) GO TO 50
      IND(IJ) = INDX
      IND(J) = IT
      IT = INDX
      KT = IX(IT)

! IF THE FIRST ELEMENT OF THE ARRAY IS GREATER THAN T,
!   INTERCHANGE IT WITH T

      INDX = IND(I)
      IF (IX(INDX) <= KT) GO TO 50
      IND(IJ) = INDX
      IND(I) = IT
      IT = INDX
      KT = IX(IT)
      GO TO 50

! INTERCHANGE ELEMENTS K AND L

40    ITT = IND(L)
      IND(L) = IND(K)
      IND(K) = ITT

! FIND AN ELEMENT IN THE UPPER PART OF THE ARRAY WHICH IS
!   NOT LARGER THAN T

50    L = L - 1
      INDX = IND(L)
      IF (IX(INDX) > KT) GO TO 50

! FIND AN ELEMENT IN THE LOWER PART OF THE ARRAY WHCIH IS NOT SMALLER THAN T

60    K = K + 1
      INDX = IND(K)
      IF (IX(INDX) < KT) GO TO 60

! IF K <= L, INTERCHANGE ELEMENTS K AND L

      IF (K <= L) GO TO 40

! SAVE THE UPPER AND LOWER SUBSCRIPTS OF THE PORTION OF THE
!   ARRAY YET TO BE SORTED

      IF (L-I > J-K) THEN
        IL(M) = I
        IU(M) = L
        I = K
        M = M + 1
        GO TO 80
      END IF

      IL(M) = K
      IU(M) = J
      J = L
      M = M + 1
      GO TO 80

! BEGIN AGAIN ON ANOTHER UNSORTED PORTION OF THE ARRAY

70    M = M - 1
      IF (M == 0) RETURN
      I = IL(M)
      J = IU(M)

80    IF (J-I >= 11) GO TO 30
      IF (I == 1) GO TO 20
      I = I - 1

! SORT ELEMENTS I+1,...,J.  NOTE THAT 1 <= I < J AND J-I < 11.

90    I = I + 1
      IF (I == J) GO TO 70
      INDX = IND(I+1)
      KT = IX(INDX)
      IT = INDX
      INDX = IND(I)
      IF (IX(INDX) <= KT) GO TO 90
      K = I

100   IND(K+1) = IND(K)
      K = K - 1
      INDX = IND(K)
      IF (KT < IX(INDX)) GO TO 100

      IND(K+1) = IT
      GO TO 90

      END SUBROUTINE ICLASS_HSJWM



*=====================================================================

      SUBROUTINE ICLASS_HOARE(N,ITAB,IRAN)
*
*     ------------------------------------------------------------------
*
*          l algorithme de tri de hoare non modifie
*                                                              h.   bung  04-88
*     ------------------------------------------------------------------
*
*             n     : dimension de itab et iran
*             itab  : tableau a classer (non modifie)
*             iran  : classement
*                     l'element de "itab" classe ieme est itab(iran(i))
*             si n .le.1 ===> iran(1)=1
*
*
      IMPLICIT NONE
*
*
      INTEGER, INTENT(IN)  :: ITAB(*),N
      INTEGER, INTENT(OUT) :: IRAN(*)
*
      INTEGER J,I,IA,IB,LPIVOT,IQ,IR,IRA,IRB,IAUX
*
*-----    variables locales pour les piles de doublets
*
      INTEGER, PARAMETER :: LGPILE=100
      INTEGER :: KOPF, IPIL1(LGPILE),IPIL2(LGPILE)
*
*
      DO  I=1,N
         IRAN(I)=I
      END DO
      IF(N < 2) RETURN
*
      J=0
      IPIL1(1)=1           !AVANT: CALL PILINI & CALL PILALL(1,N))
      IPIL2(1)=N
      KOPF=1
```

```
*
  1   CONTINUE
**avant:       call pilfre(ia,ib)
      IA=IPIL1(KOPF)
      IB=IPIL2(KOPF)
      KOPF=KOPF-1

  2   IF(IA.LT.IB) THEN
         LPIVOT=ITAB(IRAN(IA))
*--          exploration de ia+1 a ib
         IQ=IA+1
         IR=IB
   3     CONTINUE
         IF(IQ.LE.IR) THEN
             J=IR-1
             DO  I=IQ,J
                IF(ITAB(IRAN(I)).GT.LPIVOT) EXIT
             END DO
             IQ=I+1
             DO J=IR,I+1,-1
                IF(ITAB(IRAN(J)).LT.LPIVOT) EXIT
             END DO
             IAUX=IRAN(I)
             IRAN(I)=IRAN(J)
             IRAN(J)=IAUX
             IR=J-1
             GOTO 3
         ENDIF
*---         exploration terminee , on a 2 partitions a trier
         IF(I.EQ.J .AND. ITAB(IRAN(J)).LT.LPIVOT) IR=J
*--          on met le pivot a sa place definitive
         IAUX=IRAN(IA)
         IRAN(IA)=IRAN(IR)
         IRAN(IR)=IAUX
*---         on empile la partition la +grande,si elle contient 2 elts
         IRA=IR-1
         IRB=IR+1
         IF(IR-IA .GT. IB-IR) THEN
             IF(IA.LT.IRA) THEN
                 IF(KOPF >= LGPILE ) THEN
                    STOP 'ICLASS :PILES SATURéES 2'
                 ENDIF
                 KOPF=KOPF+1
                 IPIL1(KOPF)=IA
                 IPIL2(KOPF)=IRA
             ENDIF
             IA=IRB
          ELSE
             IF(IRB.LT.IB) THEN
                 IF(KOPF >= LGPILE ) THEN
                    STOP 'ICLASS :PILES SATURéES 2'
                 ENDIF
                 KOPF=KOPF+1
                 IPIL1(KOPF)=IRB
                 IPIL2(KOPF)=IB
             ENDIF
             IB=IRA
         ENDIF
         GOTO 2
      ENDIF
*
      IF(KOPF > 0 ) GOTO 1
*
      END SUBROUTINE ICLASS_HOARE
```

epx_accents.pl

```
#
# Change accented and other special characters to standard LaTeX syntax
#
$temp = "accents.tmp";
#
# Verify that temporary file is not already present
#
if (-r $temp) {
  print "File $temp already present in current directory\n";
  print "Please, rename it or delete it\n";
  exit;
}
# All arguments are file names: do the globbing
#
@FileNames = glob(join($",@ARGV));
#
foreach $name (@FileNames) {
  $base = $name;
  $base =~ s/\.ttx$//;                      # Remove '.ttx' extension if present
  $file = "$base.ttx";
  #
  # Verify that the file does exists
  #
  if ( ! -r $file ) {
    $errmsg = "File $file does not exist in current directory!\007\n";
    die "$errmsg";
  }
  print("Translating to LaTeX syntax special charracters in file $file\n");
  #
  # Process LaTeX File
  #
  open (INPUT, $file);
  open (TEMP, ">> $temp");
  while (<INPUT>) {
  #
  # Treat accented and other foreign characters
  #
    s/\xC0/\\`{A}/g; # Agrave
    s/\xC1/\\'{A}/g; # Aacute
    s/\xC2/\\^{A}/g; # Acircumflex
    s/\xC3/\\~{A}/g; # Atilde
    s/\xC4/\\"{A}/g; # Adieresis
    s/\xC5/\\AA/g;   # Aring
    s/\xC6/\\AE/g;   # AE
    s/\xC7/\\c{C}/g; # Ccedilla
    s/\xC8/\\`{E}/g; # Egrave
    s/\xC9/\\'{E}/g; # Eacute
    s/\xCA/\\^{E}/g; # Ecircumflex
    s/\xCB/\\"{E}/g; # Edieresis
    s/\xCC/\\`{I}/g; # Igrave
    s/\xCD/\\'{I}/g; # Iacute
    s/\xCE/\\^{I}/g; # Icircumflex
    s/\xCF/\\"{I}/g; # Idieresis
  #
  #fc  s/\xD0/\\xc3 /g; # Reserved
    s/\xD1/\\~{N}/g; # Ntilde
    s/\xD2/\\`{O}/g; # Ograve
    s/\xD3/\\'{O}/g; # Oacute
    s/\xD4/\\^{O}/g; # Ocircumflex
    s/\xD5/\\~{O}/g; # Otilde
    s/\xD6/\\"{O}/g; # Odieresis
  #fc  s/\xD7/\\xb0 /g; # Reserved
    s/\xD8/\\O/g;    # Oslash
    s/\xD9/\\`{U}/g; # Ugrave
    s/\xDA/\\'{U}/g; # Uacute
    s/\xDB/\\^{U}/g; # Ucircumflex
    s/\xDC/\\"{U}/g; # Udieresis
  #fc  s/\xDD/\\xc5 /g; # Reserved
  #fc  s/\xDE/\\xd7 /g; # Reserved
    s/\xDF/\\ss/g;   # germandbls
  #
    s/\xE0/\\`{a}/g; # agrave
    s/\xE1/\\'{a}/g; # aacute
    s/\xE2/\\^{a}/g; # acircumflex
    s/\xE3/\\~{a}/g; # atilde
    s/\xE4/\\"{a}/g; # adieresis
    s/\xE5/\\aa/g;   # aring
    s/\xE6/\\ae/g;   # ae
    s/\xE7/\\c{c}/g; # ccedilla
    s/\xE8/\\`{e}/g; # egrave
    s/\xE9/\\'{e}/g; # eacute
    s/\xEA/\\^{e}/g; # ecircumflex
    s/\xEB/\\"{e}/g; # edieresis
    s/\xEC/\\`{\\i}/g; # igrave
    s/\xED/\\'{\\i}/g; # iacute
    s/\xEE/\\^{\\i}/g; # icircumflex
    s/\xEF/\\"{\\i}/g; # idieresis
  #
  #fc  s/\xF0/\\xb2 /g; # Reserved
    s/\xF1/\\~{n}/g; # ntilde
    s/\xF2/\\`{o}/g; # ograve
    s/\xF3/\\'{o}/g; # oacute
    s/\xF4/\\^{o}/g; # ocircumflex
    s/\xF5/\\~{o}/g; # otilde
    s/\xF6/\\"{o}/g; # odieresis
  #fc  s/\xF7/\\xd6 /g; # Reserved
    s/\xF8/\\o/g;    # oslash
    s/\xF9/\\`{u}/g; # ugrave
    s/\xFA/\\'{u}/g; # uacute
    s/\xFB/\\^{u}/g; # ucircumflex
    s/\xFC/\\"{u}/g; # udieresis
  #fc  s/\xFD/\\xc6 /g; # Reserved
  #fc  s/\xFE/\\xca /g; # Reserved
    s/\xFF/\\"{y}/g; # ydieresis
  #
    print (TEMP);
  }
  close TEMP;
  close INPUT;
  system("del $file");
  system("ren $temp $file");
}
exit;
```

epx_autoqual.pl

```
#-------------------------------------------------------------------------------
# Automatic qualification of EUROPLEXUS input file(s)
#
#-------------------------------------------------------------------------------
# Default values
#
#-------------------------------------------------------------------------------
# There must be at least one argument (file name)
#
if ($#ARGV < 0) {                         # Index of last argument must be >= 0
  $errmsg = "Usage: epx_autoqual [-f FACTOL] name(s)[.epx]\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#       -f FACTOL   multiply all tolerances by FACTOL
#
#print "Index of last arg is: $#ARGV.\n";
$nar = $#ARGV + 1;
#print "No. of args is: $nar.\n";
while ($ARGV[0] =~ /^-/) {
  $_ = shift;
  $nar = $nar - 1;
  if (/^-f$/) {
    $_ = shift;
    $nar = $nar - 1;
    $factol = $_;
    printf "Command line switch: factol $factoln\n";
  }
  else {
    $errmsg = "ERROR: unknown switch: $_\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
}
$size_epx = $size * 14 / 9;
#print "Size of EUROPLEXUS label: $size_epx\n";
#-------------------------------------------------------------------------------
# All remaining arguments are file names: do the globbing
# If there are no names, filter all *.epx files in current directory
#
#print "No. of args is: $nar.\n";
if ($nar == 0) { # name(s) is empty
  $errmsg = "ERROR: missing file name\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
else {
  @FileNames = glob (join ($",@ARGV));
}
#-------------------------------------------------------------------------------
```

```
# Loop on file(s) to be filtered
#
foreach $name (@FileNames) {
  #-------------------------------------------------------------------------------
  # Set up file name
  #
  $base = $name;
  $base =~ s/\.epx$//;                        # Remove '.epx' extension if present
  $file = "$base";
  print "Filtering: $file\n";
  #
  # Process .epx file to generate .new file
  #
  if (defined $factol) {
    system("autoqual $file $factol");
  }
  else {
    system("autoqual $file");
  }
  #
  # Replace .epx file by .new file
  #
  system("mv -f $file.new $file.epx");
}
exit;

#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_batch.pl

```
#-------------------------------------------------------------------------------
# Run EUROPLEXUS in batch mode (no login!) via the at command
#-------------------------------------------------------------------------------
# Redirect STDOUT and STDERR to epx_batch.log (Fixed path!!!!)
#
$logfile = "E:\\EUROPLEXUS\\epx_batch.log";
open STDOUT, ">>$logfile";
open STDERR, ">>$logfile";
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 0
#
if ($#ARGV >= 0) {                          # Index of last argument must be = -1
  $errmsg = "Usage: epx_batch\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#
#-------------------------------------------------------------------------------
# Set and check existence of run directory
#
$rundir = "E:\\CEA\\Linux\\JRC";                      # Directory on which to run
if ( ! -d $rundir) {
  die "The EUROPLEXUS directory: $rundir does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Make sure we are in $rundir (so we may use . instead of $rundir)
#
$curdir = `pwd`;
chop $curdir;
if ( $curdir ne $rundir ) {
  print "Changing directory to $rundir.\n";
  chdir "$rundir" ||
  die "Can't chdir to $rundir!\007\n";
}
#-------------------------------------------------------------------------------
# Run test case
#
system ("epx_bench -l fin_pulse_3d");
#
exit;


#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_bench.pl

```
#-------------------------------------------------------------------------------
# Execute EUROPLEXUS benchmark test
#-------------------------------------------------------------------------------
# Modules needed
#
use HTTP::Date;
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$copy = "cp -p";
$del = "rm -f";
$errfil = "epx_bench.err";
#-------------------------------------------------------------------------------
# Check number of arguments, must be from 1 to 5
#
if ($#ARGV < 0 || $#ARGV > 5) {     # Index of last argument must be from 0 to 5
  $errmsg = "Usage: epx_bench [-e <exec>[.exe]] [-c] [-w] [-l] [-b] [-M nn] base
name[.epx]\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#       -e     use exectutable <exec>[.exe] instead of standard executable
#       -c     use -check version of standard executable
#       -w     use standard Quickwin exectutable instead of standard executable
#       -l     local, i.e. use local input file, not from the bench directory
#       -b     launch code in batch mode, not in interactive mode.
#              The STDOUT is redirected. The CONV directive should NOT be
#              present in the input file.
#       -M     launch MPI version of the code. The file mpiexec.exe must be stor
ed in
#              C:\Program Files\MPICH2\bin\; nn is the number of processors used
.
#
while ($ARGV[0] =~ /^-/) {
  $_ = shift;
  if (/^-l(.*)/) {
    $local = "yes";
  }
  elsif (/^-e$/) {
    $_ = shift;
    $exe = $_;
  }
  elsif (/^-c$/) {
    $check = "yes";
    printf "Using -check version of the standard executable\n";
  }
  elsif (/^-w$/) {
    $qw = "QuickWin";
  }
  elsif (/^-b$/) {
    $batch = "yes";
  }
  elsif (/^-M$/) {
    $_ = shift;
    $mpi_nn = $_;
    $mpi_exe="\"C:\\Program Files\\MPICH2\\bin\\mpiexec.exe\"";
    if ( $mpi_nn == 0 ) {
      $errmsg = "The number of processors must be higher than 0!\007\n";
      &ERRFIL ($errfil, $errmsg); die "$errmsg";
    }
  }
  else {
    $errmsg = "ERROR: unknown switch: $_\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  $errmsg = "The EUROPLEXUS environment variable is undefined!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS bench sub-directory exists
#
if (! defined ($local)) {
  $epxb = "$epx\\bench";
  if ( ! -d $epxb) {
    $errmsg = "The EUROPLEXUS bench directory: $epxb does not exist!\007\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
}
else {
  $epxb = "\.";
}
#-------------------------------------------------------------------------------
# Choose the standard executable, if -e <exec> has not been specified
#
if ( ! defined $exe) {
  #
  # Copy the standard executable to the %TEMP% directory under a unique name
  #
  # Check that the temporary directory exists
  $temp = $ENV{'TEMP'};
  if ( ! -d $temp) {
    $errmsg = "The EUROPLEXUS temporary directory: $temp does not exist!\007\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
  #
  if ( defined $mpi_nn ) {
    $stdexe = "$epx\\exe\\europlexus_mpi.exe";
  }
  elsif ( defined $check ) {
    $stdexe = "$epx\\exe\\europlexus_check.exe";
  }
  elsif ( defined $qw ) {
    $stdexe = "$epx\\exe\\europlexusqw.exe";
  }
  else {
    $stdexe = "$epx\\exe\\europlexus.exe";
  }
  # Get current date and time
  #
  $time = time;                       # Machine time (seconds since epoch)
  $gmttime = time2str($time);         # Universal (GMT) date and time
  $gmttime =~ s/.....//;
  $gmttime =~ s/ GMT//;
  $gmttime =~ s/ /_/;
  $gmttime =~ s/ /_/;
  $gmttime =~ s/ /_/;
  $gmttime =~ s/:/_/;
  $gmttime =~ s/:/_/;
  $tempexe = "$temp\\epx_$gmttime.exe";
  system("$copy $stdexe $tempexe");
```

```
  $exe = $tempexe;
}
#-------------------------------------------------------------------------------
# Check the chosen executable
#
$ebase = $exe;
$ebase =~ s/\.exe$//;
$exe = "$ebase\.exe";
#
if ( ! -x $exe ) {
  $errmsg = "The EUROPLEXUS executable: $exe is not executable!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# The argument is the (base) name of the test
#
$name = $ARGV[0];
#-------------------------------------------------------------------------------
# Set up file name
#
$base = $name;
$base =~ s/\.epx$//;                        # Remove '.epx' extension if present
#-------------------------------------------------------------------------------
# Verify that the input file exists in the appropriate dir
#
$inp = "$base.epx";
$msh = "$base.msh";
$epxinp = "$epxb\\$inp";
$epxmsh = "$epxb\\$msh";
if ( ! -r $epxinp ) {
  $errmsg = "File $inp does not exist in directory $epxb!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Copy input and mesh (if any) files from the EPX bench dir to current dir,
# if necessary
#
if (! defined ($local)) {
  system("$copy $epxinp .");
  if ( -r $epxmsh ) {
    system("$copy $epxmsh .");
  }
}
#-------------------------------------------------------------------------------
# Prepare (copy) files for run execution
#
system("$del $base.std");
system("$del fort.*");
#system("$copy $base.epx fort.15");
if ( -r $msh ) {
  system("$copy $base.msh fort.9");
}
#-------------------------------------------------------------------------------
# Execute the run
#
if (! defined ($batch)) {
  print "Running $exe in interactive mode on benchmark: $base\n";
# system("$exe $base 2>$base.std");
  if( defined ($mpi_nn)) {
    system("$mpi_exe -n $mpi_nn -noprompt $exe $base");
  }
  else {
    system("$exe $base");
  }
}
else {
  print "Running $exe in batch mode on benchmark: $base\n";
# system("$exe $base >$base.eco 2>$base.std");
  if( defined ($mpi_nn)) {
    system("$mpi_exe -n $mpi_nn -noprompt $exe $base >$base.eco");
  }
  else {
    system("$exe $base >$base.eco");
  }
}
#-------------------------------------------------------------------------------
# Clean up: delete or rename fort.*
#
if (-r $msh) {unlink "fort.9"};
#unlink "fort.15";
#if (-r "fort.16") {rename ("fort.16", "$base.listing")};
#if (-r "fort.24") {rename ("fort.24", "$base.ps")};
#
# Delete the temporary executable if any
if (defined $tempexe) {
  sleep(1); # allow some time for the system to free the executable ...
  unlink $tempexe;
  if (-r $tempexe ) {
    print "Oops $tempexe is still there!!!! I try now with rm -f ...\007\n";
    system ("rm -f $tempexe");
    if (-r $tempexe ) {
      print "ATTENTION, $tempexe is still there!!!! Delete it manually!\007\n";
    }
  }
}
#-------------------------------------------------------------------------------
# Validate the run results
#
unlink "epx_vali.err";
system("epx_vali $base");
if ( -r "epx_vali.err" ) {
  $errmsg = "Wrong validation for $base!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#
exit;


#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_bmpinset.pl

```
#-------------------------------------------------------------------------------
# Inset in all .BMP files base_<nnnn> the pict.bmp, producing inset_<nnnn>
# on the current directory. The pict file must be smaller than the base.
#-------------------------------------------------------------------------------
system ("rm -f inset_*.bmp");
$pict = "pict.bmp";
if (! -r $pict) {
  die "The picture file $pict is missing!\007\n";
}
system ("cp -f $pict _pict.bmp");
@filenames = `ls base_*.bmp`;
foreach $basefile (@filenames) {
  $basefile =~ s/\n$//;
# print "Base file : $basefile";
  $num = $basefile;
  $num =~ s/base_//;
  $num =~ s/\.bmp//;
# print ", Num       : $num";
  $inset = "inset_$num.bmp";
  print "Inserting in file $basefile the picture $pict --> $inset\n";
  system ("cp -f $basefile _base.bmp");
  system ("bmpinset >_tmp.txt 2>&1");
  system ("mv _inset.bmp $inset");
  system ("rm -f _base.bmp");
}
system ("rm -f _pict.bmp");
exit;
```

epx_bmpjoin.pl

```
#-------------------------------------------------------------------------------
# Join all .BMP file couples left_<nnnn> right_<nnnn> into join_<nnnn>
# on the current directory. Each couple of files must have same height.
#-------------------------------------------------------------------------------
system ("rm -f join_*.bmp");
@filenames = `ls left_*.bmp`;
foreach $left (@filenames) {
  $left =~ s/\n$//;
# print "Left file : $left";
  $num = $left;
  $num =~ s/left_//;
  $num =~ s/\.bmp//;
# print ", Num       : $num";
  $right = "right_$num.bmp";
# print ", Right file : $right\n";
  $join = "join_$num.bmp";
  print "Joining files $left and $right --> $join\n";
  if (! -r $right) {
    die "The right file $right is missing!\007\n";
  }
  system ("cp -f $left _left.bmp");
  system ("cp -f $right _right.bmp");
  system ("bmpjoin >_tmp.txt 2>&1");
  system ("mv _join.bmp $join");
  system ("rm -f _left.bmp");
  system ("rm -f _right.bmp");
}
exit;
```

epx_checkat.bat

```
schtasks
schtasks /query /fo LIST /v
```

epx_clean.pl

```
#
# Clean up results of EUROPLEXUS benchmark test
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 1
#
if ($#ARGV != 0) {                          # Index of last argument must be = 0
  die "Usage: epx_clean basename[.epx]\007\n";
}
#-------------------------------------------------------------------------------
# The argument is the (base) name of the test
#
$name = $ARGV[0];
#-------------------------------------------------------------------------------
# Set up file name
#
$base = $name;
$base =~ s/\.epx$//;                        # Remove '.epx' extension if present
print "Cleaning up benchmark: $base\n";
#-------------------------------------------------------------------------------
# Delete the test files
#
system("del $base.*");
system("del fort.*");
#-------------------------------------------------------------------------------
# Delete the compar files
#
system("del coco.txt psps.txt");
#
exit;
```

epx_clean_manual_files.pl

```
#-------------------------------------------------------------------------------
$del  = "rm -f";
#-------------------------------------------------------------------------------
system ("$del gbnotice.tex");
system ("$del gbnotice.log");
system ("$del gbnotice.aux");
system ("$del gbnotice.idx");
```

```
system ("$del gbnotice.toc");
system ("$del gbnotice.ilg");
system ("$del gbnotice.ind");
system ("$del gbnotice.out");
system ("$del gbnotice.tid");
system ("$del gbnotice.dvi");
system ("$del gbnotice.ps");
system ("$del gbnotice.html");
system ("$del gbnotice.pdf");
#-------------------------------------------------------------------------------
exit;
```

epx_cmp.pl

```
#-------------------------------------------------------------------------------
# Compile Fortran source file(s) of EUROPLEXUS
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$optimize = "/optimize:0 /debug:full";    # By default, debugging version
#$optimize = "$optimize /debug-parameters:all /Qtrapuv";
$optimize = "$optimize /debug-parameters:all";
$errfil = "epx_cmp.err";
$keys = "WIN OGL W32 SPLIB MFFT MKL";
$keys2 = " ";
$blan = " ";
$ordo = "epx_ordo.txt";
$libs = "";
$errs = 0;
$inc_mpi = "";
$modu = "module";
$mkl = "/Qmkl:sequential"; # By default, sequential MKL libraries
#-------------------------------------------------------------------------------
# Any number of arguments is accepted (including 0)
#
if ($#ARGV < -1) {                        # Index of last argument must be >= -1
  $errmsg = "Usage: epx_cmp [-q] [-o] [-w] [-k KEY] [-x] [-g] [-c] [-C] [-M] [-n
omkl] [-times] [name(s)[.ff]]\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#       -q       quiet (do not print subsequent switch echo)
#       -o       optimize
#       -w       full warnings, and treated as errors
#       -k KEY   add key KEY to filtering keys
#       -x       suppress all default filtering keys
#       -g       use graphical libraries (QuickWin) : necessary to compile
#                all program units in current dir to produce QuickWin version
#       -c       check all at run-time
#       -C       check all at run-time + floating point checks
#       -M       compile for MPI (parallel) version
#       -nomkl   do not use Intel MKL libraries (use local Blas/Lapack)
#       -times   print starting and ending times of each compilation
#                in the corresponding .err file for subsequent processing
#
# Note : if name(s) is missing, then compile all *.ff files in the current
#        directory. To find the correct compilation order, use epx_ordo.
#
#print "Index of last arg is: $#ARGV.\n";
$nar = $#ARGV + 1;
#print "No. of args is: $nar.\n";
if ($#ARGV >= 0) { # There is at least one argument
  while ($ARGV[0] =~ /^-/) {
    $_ = shift;
    $nar = $nar - 1;
    if (/^-q(.*)/) {
      $quiet = "yes";
    }
    elsif (/^-o(.*)/) {
      $optimize = "/optimize:5 /nodebug";
      if (! defined ($quiet)) {printf "Command line switch: optimize\n";}
    }
    elsif (/^-w(.*)/) {
      $optimize = "/optimize:0 /debug:full /warn:all";
      $optimize = "$optimize /warn:errors /noerror-limit";
#     $optimize = "$optimize /debug-parameters:all /Qtrapuv";
      $optimize = "$optimize /debug-parameters:all";
      if (! defined ($quiet)) {printf "Command line switch: warnings\n";}
    }
    elsif (/^-c(.*)/) {
      $check = "yes";
      $optimize = "$optimize /check:all /check:noarg_temp_created";
      if (! defined ($quiet)) {printf "Command line switch: check\n";}
    }
    elsif (/^-C(.*)/) {
      $check = "yes";
      $checkfp = "yes";
      $optimize = "$optimize /check:all";
      if (! defined ($quiet)) {printf "Command line switch: check + floating poi
nt\n";}
    }
    elsif (/^-k$/) {
      $_ = shift;
      $nar = $nar - 1;
      $keys2 = $keys2.$_.$blan;
      printf "Command line switch: additional filter keys $keys2\n";
    }
    elsif (/^-x(.*)/) {
      printf "Command line switch: suppress default filter keys $keys\n";
      $keys = " ";
    }
    elsif (/^-g(.*)/) {
      printf "Command line switch: use graphical libraries (QuickWin)\n";
      $libs = "/libs:qwin";
    }
    elsif (/^-M$/) {
      $mpi = "yes";
      $keys2 = $keys2.'MPI'.$blan;
      $modu = "module_mpi"; # Use MPI version of EPX module files
      printf "Command line switch: compile for MPI\n";
    }
    elsif (/^-nomkl$/) {
      $mkl = "";
      printf "Command line switch: do not use Intel MKL libraries\n";
    }
    elsif (/^-times$/) {
      $CPU = "yes";
      printf "Command line switch: measure compilation times\n";
    }
    else {
      $errmsg = "ERROR: unknown switch: $_\n";
      &ERRFIL ($errfil, $errmsg); die "$errmsg";
    }
  }
}
#print "Index of last arg is: $#ARGV.\n";
#print "No. of args is: $nar.\n";
#-------------------------------------------------------------------------------
#
$keys = $keys.$keys2;
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  $errmsg = "The EUROPLEXUS environment variable is undefined!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Now that $epx is set, we can set also $inc_mpi if needed
if (defined $mpi) {
  $inc_mpi = "/include:$epx\\include_mpi"; # Add MPI include folder
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  $errmsg = "The EUROPLEXUS directory: $epx does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# All remaining arguments are file names: do the globbing
# If there are no names, compile all *.ff files in the order found
# by running epx_ordo.
#
#print "No. of args is: $nar.\n";
if ($nar == 0) { # name(s) is empty
  print "Compile all current sources in epx_ordo order!\n";
  system ("epx_ordo");
  #-----------------------------------------------------------------------------
  # Build array of source file names from file epx_ordo.txt
  #
  open (ORDO, $ordo);
  while (<ORDO>) {
    chop;
    $fil = $_;
    if ( ! -r $fil ) {
      $errmsg = "Died epx_cmp: $fil missing!\007\n";
      &ERRFIL ($errfil, $errmsg); die "$errmsg";
    }
    push @FileNames, $_;
  }
  close ORDO;
}
else {
  @FileNames = glob (join ($",@ARGV));
}
#-------------------------------------------------------------------------------
# Loop on file(s) to be compiled
#
foreach $name (@FileNames) {
  #-----------------------------------------------------------------------------
  # Set up file name
  #
  $base = $name;
  $base =~ s/\.ff$//;                       # Remove '.ff' extension if present
  $file = "$base.ff";
  print "Filtering and compiling: $file\n";
  #-----------------------------------------------------------------------------
  # Verify that the file does exists
  #
  if ( ! -r $file ) {
    $errmsg = "File $file does not exist in current directory!\007\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
  #-----------------------------------------------------------------------------
  # Filter the .ff file to obtain the .f
  #
  open (ERR, "> $base.err");
  print ERR "Filtering file $base.ff with keys: $keys\n";
  print ERR "============================================================\n";
  #
  if (! system ("epx_filter $keys <$file >$base.f 2>_devnull")) {
    $errmsg = "Filtering ERROR(S) in $file! (See $base.err & $base.f)\007\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
  #-----------------------------------------------------------------------------
  # Compile the .f file
  #
  $inc = "/include:$epx\\include";          # Where to search for includes (.inc)
  $mod = "/include:$epx\\$modu";            # Where to search for modules (.mod)
  #
  # Special case for -C option : if the file to be compiled is "main"
  # then add the switches "/fpe:0 /fp:strict"
  #
  if ($base eq "main") {
    if (defined $checkfp) {
      printf "Compilation with -C of file main.ff : add /fpe:0 /
fp:strict switches!!!\n";
      $optimize = "$optimize /fpe:0 /fp:strict";
    }
  }
  #
  $opt = "$mkl /compile-only $inc $inc_mpi $mod $optimize $libs /automatic /
traceback";
  #
  if (defined $CPU) {
    $time1 = time;                        # Machine time (seconds since epoch)
```

```
    print ERR "Starting time: $time1\n";
  }
  print ERR "Compiling file $base.f with options:\n";
  print ERR "$opt\n";
  print ERR "============================================================\n";
  close ERR;
  #
  if (  system ("ifort $opt $base.f >>$base.err 2>&1")) {
    $errmsg = "Compilation ERROR(S) in $base.f! (See file $base.err)\007\n";
    &ERRFIL ($errfil, $errmsg);
    $errs = $errs + 1;
    print "Compilation ERROR(S) in $base.f! (See file $base.err)\007\n";
  }
  if (defined $CPU) {
    open (ERR, ">> $base.err");
    $time2 = time;                      # Machine time (seconds since epoch)
    print ERR "Ending time  : $time2\n";
    close ERR;
  }
}
#
if ($errs > 0) {
  $errmsg = "Died epx_cmp: $errs compilation failures(s)!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
else {
  print "Sources compilation has successfully terminated.\n";
}
#
exit;


#------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#------------------------------------------------------------------------------
```

epx_cmplst.bat

```
@rem = '--*-Perl-*--
@echo off
if "%OS%" == "Windows_NT" goto WinNT
perl -x -S "%0" %1 %2 %3 %4 %5 %6 %7 %8 %9
goto endofperl
:WinNT
perl -x -S %0 %*
if NOT "%COMSPEC%" == "%SystemRoot%\system32\cmd.exe" goto endofperl
if %errorlevel% == 9009 echo You do not have Perl in your PATH.
if errorlevel 1 goto script_failed_so_exit_with_non_zero_val 2>nul
goto endofperl
@rem ';
#!perl
#line 15
#------------------------------------------------------------------------------
# Compare EUROPLEXUS listing(s) with standard one(s)
#------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#------------------------------------------------------------------------------
# Default values
#
#------------------------------------------------------------------------------
# Check number of arguments, must be >= 1
#
if ($#ARGV < 0) {                       # Index of last argument must be >= 0
  $errmsg = "Usage: epx_cmplst name(s)[.listing]\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  $errmsg = "The EUROPLEXUS environment variable is undefined!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  $errmsg = "The EUROPLEXUS directory: $epx does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#------------------------------------------------------------------------------
# All remaining arguments are file names: do the globbing
#
@FileNames = glob (join ($",@ARGV));
#------------------------------------------------------------------------------
# Loop on file(s) to be compiled
#
foreach $name (@FileNames) {
    $base = $name;
    $base =~ s/\.listing$//;            # Remove '.listing' extension if present
    $file = "$base.listing";
    print("\n\n#########>Comparing $file\n");
    system("diff C:\\EUROPLEXUS\\Bench\\$file $file");
}

__END__
:endofperl
```

epx_cmplst.pl

```
#------------------------------------------------------------------------------
# Compare EUROPLEXUS listing(s) with standard one(s)
#------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#------------------------------------------------------------------------------
# Default values
#
#------------------------------------------------------------------------------
# Check number of arguments, must be >= 1
#
if ($#ARGV < 0) {                       # Index of last argument must be >= 0
  $errmsg = "Usage: epx_cmplst name(s)[.listing]\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  $errmsg = "The EUROPLEXUS environment variable is undefined!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  $errmsg = "The EUROPLEXUS directory: $epx does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#------------------------------------------------------------------------------
# All remaining arguments are file names: do the globbing
#
@FileNames = glob (join ($",@ARGV));
#------------------------------------------------------------------------------
# Loop on file(s) to be compiled
#
foreach $name (@FileNames) {
    $base = $name;
    $base =~ s/\.listing$//;            # Remove '.listing' extension if present
    $file = "$base.listing";
    print("\n\n#########>Comparing $file\n");
    system("diff C:\\EUROPLEXUS\\Bench\\$file $file");
}
```

epx_comparisons.pl

```
#
# Change comparison operators from F77 to F90 syntax
#
$temp = "comparisons.tmp";
#
# Verify that temporary file is not already present
#
if (-r $temp) {
  print "File $temp already present in current directory\n";
  print "Please, rename it or delete it\n";
  exit;
}
# All arguments are file names: do the globbing
#
@FileNames = glob(join($",@ARGV));
#
foreach $name (@FileNames) {
  $base = $name;
  $base =~ s/\.ff$//;                       # Remove '.ff' extension if present
  $file = "$base.ff";
  #
  # Verify that the file does exists
  #
  if ( ! -r $file ) {
    $errmsg = "File $file does not exist in current directory!\007\n";
    die "$errmsg";
  }
  print("Translating to F90 syntax comparison operators in file $file\n");
  #
  # Process Fortran File
  #
  open (INPUT, $file);
  open (TEMP, ">> $temp");
  while (<INPUT>) {
  #
    s/\.EQ\./ == /g;
    s/\.NE\./ \/= /g;
    s/\.GT\./ > /g;
    s/\.LT\./ < /g;
    s/\.GE\./ >= /g;
    s/\.LE\./ <= /g;
#
    s/\.eq\./ == /g;
    s/\.ne\./ \/= /g;
    s/\.gt\./ > /g;
    s/\.lt\./ < /g;
    s/\.ge\./ >= /g;
    s/\.le\./ <= /g;
  #
    print (TEMP);
  }
  close TEMP;
  close INPUT;
  system("del $file");
  system("ren $temp $file");
}
exit;
```

epx_correct_ps.pl

```
#------------------------------------------------------------------------------
# Correct line thickness and font size in PostScript file generated by
# EUROPLEXUS
#
#------------------------------------------------------------------------------
# Default values
#
$temp = "_temp.ps";
```

```
$thick = "1.0";
$size = "10";
#-------------------------------------------------------------------------------
# There must be at least one argument (file name)
#
if ($#ARGV < 0) {                       # Index of last argument must be >= 0
  $errmsg = "Usage: epx_correct_ps [-t THICK] [-s SIZE] name(s)[.ps]\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#       -t THICK    use line thickness THICK (in points)
#       -s SIZE     use font size SIZE (in points)
#
#print "Index of last arg is: $#ARGV.\n";
$nar = $#ARGV + 1;
#print "No. of args is: $nar.\n";
while ($ARGV[0] =~ /^-/) {
  $_ = shift;
  $nar = $nar - 1;
  if (/^-t$/) {
    $_ = shift;
    $nar = $nar - 1;
    $thick = $_;
    printf "Command line switch: line thickness $thick\n";
  }
  elsif (/^-s$/) {
    $_ = shift;
    $nar = $nar - 1;
    $size = $_;
    printf "Command line switch: line size $size\n";
  }
  else {
    $errmsg = "ERROR: unknown switch: $_\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
}
$size_epx = $size * 14 / 9;
#print "Size of EUROPLEXUS label: $size_epx\n";
#-------------------------------------------------------------------------------
# All remaining arguments are file names: do the globbing
# If there are no names, filter all *.ps files in current directory
#
#print "No. of args is: $nar.\n";
if ($nar == 0) { # name(s) is empty
  $errmsg = "ERROR: missing file name\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
else {
  @FileNames = glob (join ($",@ARGV));
}
#-------------------------------------------------------------------------------
# Verify that temporary file is not already present
#
if (-r $temp) {
  print "File $temp already present in current directory\n";
  print "Please, rename it or delete it\n";
  exit;
}
#-------------------------------------------------------------------------------
# Loop on file(s) to be filtered
#
foreach $name (@FileNames) {
  #-----------------------------------------------------------------------------
  # Set up file name
  #
  $base = $name;
  $base =~ s/\.ps$//;                       # Remove '.ps' extension if present
  $file = "$base.ps";
  print "Filtering: $file\n";
  #
  # Process PostScript File
  #
  open (INPUT, "type $file |");
  open (TEMP, ">> $temp");
  while (<INPUT>) {
    s/0.1 setlinewidth/$thick setlinewidth/;
    s/0.10 setlinewidth/$thick setlinewidth/;
    s/Courier findfont 9/CourierBold findfont $size/;
    s/Courier findfont 14/CourierBold findfont $size_epx/;
    print (TEMP);
  }
  close TEMP;
  close INPUT;
  system("del $file");
  system("ren $temp $file");
}
exit;

#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_customize_fortran.pl

```
$fromdir = '"\\\\sm61\c\Program Files\Intel\Compiler\Fortran\10.0.025\IA32\Inclu
de"';
#print "Fromdir:$fromdir\n";
$todir = '"C:\Program Files\Intel\Compiler\Fortran\10.0.025\IA32\Include"';
#print "Todir:$todir\n";
system("scopy $fromdir\\opengl_fwrap.mod $todir");
system("scopy $fromdir\\opengl_gl.mod $todir");
system("scopy $fromdir\\opengl_glinterfaces.mod $todir");
system("scopy $fromdir\\opengl_glu.mod $todir");
system("scopy $fromdir\\opengl_gluinterfaces.mod $todir");
system("scopy $fromdir\\opengl_glut.mod $todir");
system("scopy $fromdir\\opengl_glutinterfaces.mod $todir");
system("scopy $fromdir\\opengl_kinds.mod $todir");

$fromdir = '"\\\\sm61\c\Program Files\Intel\Compiler\Fortran\10.0.025\IA32\Lib"'
;
#print "Fromdir:$fromdir\n";
$todir = '"C:\Program Files\Intel\Compiler\Fortran\10.0.025\IA32\Lib"';
#print "Todir:$todir\n";
system("scopy $fromdir\\bmplib.lib $todir");
system("scopy $fromdir\\f90GL.lib $todir");
system("scopy $fromdir\\f90GLU.lib $todir");
system("scopy $fromdir\\f90GLUT.lib $todir");

$fromdir = '"\\\\sm61\c\Program Files\Microsoft Visual Studio 8\Vc\PlatformSDK\I
nclude\gl"';
#print "Fromdir:$fromdir\n";
$todir = '"C:\Program Files\Microsoft Visual Studio 8\Vc\PlatformSDK\Include\gl"
';
#print "Todir:$todir\n";
system("scopy $fromdir\\glut.h $todir");
system("scopy $fromdir\\glutf90.h $todir");

$fromdir = '"\\\\sm61\c\Program Files\Microsoft Visual Studio 8\Vc\PlatformSDK\L
ib"';
#print "Fromdir:$fromdir\n";
$todir = '"C:\Program Files\Microsoft Visual Studio 8\Vc\PlatformSDK\Lib"';
#print "Todir:$todir\n";
system("scopy $fromdir\\Glut32.lib $todir");
```

epx_deobso.pl

```
#-------------------------------------------------------------------------------
# Restore previously set aside (obsolete) file(s)
#-------------------------------------------------------------------------------
# Default values
#
$del = "rm -f";
$move = "mv -f";
$ext = "ff";                                # By default, source file
$dir = "source";                            # By default, source directory
$sour = 1;                                  # By default, it is a source file
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 1or 2
#
if ($#ARGV > 1) {                       # Index of last argument must be = 0 or 1
  die "Usage: epx_deobso [-i] [-b] [-m] name[.<extension>]\007\n";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#       -i     include file, i.e. ".inc" file, not source (.ff) file
#       -b     benchmark, i.e. ".epx" file, not source (.ff) file
#       -m     manual, i.e. ".ttx" file, not source (.ff) file
#
while ($ARGV[0] =~ /^-/) {
  $_ = shift;
  if (/^-i(.*)/) {
    $ext = "inc";
    $dir = "include";
    undef $sour;
    printf "Command line switch: include file\n";
  }
  elsif (/^-b(.*)/) {
    $ext = "epx";
    $dir = "bench";
    undef $sour;
    $ben = 1;
    printf "Command line switch: bench file\n";
  }
  elsif (/^-m(.*)/) {
    $ext = "ttx";
    $dir = "manual";
    $manfil = "yes";
    undef $sour;
    printf "Command line switch: manual file\n";
  }
  else {
    die "ERROR: unknown switch: $_\n";
  }
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  die "The EUROPLEXUS environment variable is undefined!\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directories exist
#
if ( ! -d $epx ) {
  die "The EUROPLEXUS directory: $epx does not exist!\007\n";
}
#
$hisdir = "$epx\\history";
if ( ! -d $hisdir ) {
  die "The history directory: $hisdir does not exist!\007\n";
}
#
$hisobs = "$hisdir\\obsolete";
if ( ! -d $hisobs ) {
  die "The history obsolete directory: $hisobs does not exist!\007\n";
}
#
$moddir = "$epx\\module";
if ( ! -d $moddir ) {
  die "The module directory: $moddir does not exist!\007\n";
}
#
$objlib = "$epx\\library\\libplex.lib";
if ( ! -r $objlib ) {
  die "The object library: $objlib does not exist!\007\n";
}
#
$manfildir = "$epx\\manual_filtered";
if ( ! -d $manfildir ) {
  die "The filtered manual directory: $manfildir does not exist!\007\n";
}
#
$obsodir = "$epx\\obsolete";
if ( ! -d $obsodir ) {
  system ("mkdir $obsodir");
```

```
}
#-------------------------------------------------------------------------------
# Set up file name
#
$name = $ARGV[0];                           # Get fist and only argument
$base = $name;
$base =~ s/\.$ext$//;                       # Remove extension if present
$file = "$base.$ext";
#-------------------------------------------------------------------------------
# Verify that the file exists in the corresponding directory
#
$fromdir = "$epx\\$dir\\obsolete";          # Full directory name
$from = "$fromdir\\$file";                  # Full file name
if ( ! -r $from ) {
  die "File $from does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Target directory and target file
#
$todir =  "$epx\\$dir";                     # Full directory name
if ( ! -d $todir ) {
  die "The target directory: $todir does not exist!\007\n";
}
$to = "$todir\\$file";                      # Full file name
if ( -r $to ) {
  printf "Removing $to\n";
  system("$del $to");
}
printf "Moving $from $to\n";
system("$move $from $to");
#-------------------------------------------------------------------------------
# Additional treatment for bench files : move also auxiliary files
#
if ( defined $ben ) {
  $fromfiles = "$fromdir\\$base.\*";
  printf "Moving $fromfiles $todir\n";
  system("$move $fromfiles $todir");
}
#-------------------------------------------------------------------------------
# Treat also corresponding history file, if any
#
$fromhis = "$hisobs\\$base.his";
if ( -r $fromhis ) {
  $tohis = "$hisdir\\$base.his";            # Full file name
  if ( -r $tohis ) {
    printf "Removing $tohis\n";
    system("$del $tohis");
  }
  printf "Moving $fromhis $tohis\n";
  system("$move $fromhis $tohis");
}
#-------------------------------------------------------------------------------
# Chdir to work directory $obsodir
$curdir = `pwd`;
chop $curdir;
if ( $curdir ne $obsodir ) {
  print "\nChanging directory to $obsodir.\n";
  chdir "$obsodir" ||
  die "Can't chdir to $obsodir!\007\n";
}
#-------------------------------------------------------------------------------
# Additional treatment for source files : restore corresponding object file
# into the library, and also restore corresponding .mod file if any
#
if ( defined $sour ) {
  printf "Restoring $base.obj into $objlib\n";
  system("epx_get $base");
  system("epx_cmp -o $base");
  system("LIB $objlib $base.obj");
  $mod = "$base.mod";               # Full file name
  if ( -r $mod ) {
    printf "Restoring $mod\n";
    $tomod = "$moddir\\$base.mod";
    system("$move $mod $tomod");
  }
}
#-------------------------------------------------------------------------------
# Additional treatment for manual files : restore corresponding filtered file
# into the manual_filtered directory
#
if ( defined $manfil ) {
  printf "Restoring $base.tex into $manfildir\n";
  system("epx_get -m $base");
  system("epx_filter_manual $base");
  $tex = "$base.tex";                           # Full file name
  $totex = "$manfildir\\$base.tex";             # Full file name
  if ( -r $tex ) {
    printf "Restoring $tex\n";
    system("$move $tex $totex");
  }
}
#-------------------------------------------------------------------------------
# Clean up
#-------------------------------------------------------------------------------
system("$del $base.*");
#-------------------------------------------------------------------------------
exit;
```

epx_diff.pl

```
#-------------------------------------------------------------------------------
# Diff local Fortran source|include file against same from EUROPLEXUS library
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$ext = "ff";                            # By default, source file
$dir = "source";                        # By default, source directory
$cmd = "diff";                          # By default, diff
$optlist = "0";                         # By default 0
#-------------------------------------------------------------------------------
# Check number of arguments, must be 1 or 2 or 3
#
if ($#ARGV < 0) {                       # Index of last argument must be >= 0
  die "Usage: epx_diff [-i] [-s] [-b] [-m] [-c] [-l] [-p] [-u] [-v] [-w] [-d] [-
a] [-g] [-f] [-?] name(s)[.<extension>]\007\n";
}

#-------------------------------------------------------------------------------
# Process optional switches:
#       -i     include file, i.e. ".inc" file, not source (.ff) file
#       -s     side-by-side comparison
#       -b     benchmark, i.e. ".epx" file, not source (.ff) file
#       -m     manual, i.e. ".ttx" file, not source (.ff) file
#       -c     ignore case (uppercase and lowercase are considered equal)
#       -l     listing file, i.e. ".listing" file, not source (.ff) file
#       -p     postscript file, i.e. ".ps" file, not source (.ff) file
#       -u     utility file, i.e. ".pl" file, not source (.ff) file
#       -v     validation file, i.e. ".vld" file, not source (.ff) file
#       -w     ignore all white space
#       -d     specify directory for reference files, instead of having
#              it determined by the file suffix. If used, it must be
#              the last switch (i.e. after -ibmlp)
#       -a     animation file, i.e. ".avi" file, not source (.ff) file
#       -g     graphics file, i.e. ".bmp" file, not source (.ff) file
#       -f     only the files with differences are displayed

# NOTE : -s and -l are probably incompatible!
#
while ($ARGV[0] =~ /^-/) {
  $_ = shift;

  if (/^-\?(.*)/) {
     printf "Diff local Fortran source|include file against same from EUROPLEXUS
 library\n";
     printf "Process optional switches:\n";
     printf " -i   include file, i.e. \".inc\" file, not source (.ff) file\n";
     printf " -s   side-by-side comparison\n";
     printf " -b   benchmark, i.e. \".epx\" file, not source (.ff) file\n";
     printf " -m   manual, i.e. \".ttx\" file, not source (.ff) file\n";
     printf " -c   ignore case (uppercase and lowercase are considered equal)\n"
;
     printf " -l   listing file, i.e. \".listing\" file, not source (.ff) file\n
";
     printf " -p   postscript file, i.e. \".ps\" file, not source (.ff) file\n";
     printf " -u   utility file, i.e. \".pl\" file, not source (.ff) file\n";
     printf " -v   validation file, i.e. \".vld\" file, not source (.ff) file\n"
;
     printf " -w   ignore all white space\n";
     printf " -d   specify directory for reference files, instead of having it\n
";
     printf "      determined by the file suffix. If used, it must be the\n";
     printf "      last switch (i.e. after -ibmlp)\n";
     printf " -a   animation file, i.e. \".avi\" file, not source (.ff) file\n";
     printf " -g   graphics file, i.e. \".bmp\" file, not source (.ff) file\n";
     printf " -f   only the files with differences are displayed\n";
     printf " -?   print this help\n";
     printf "NOTE : -s and -l are probably incompatible!\n";
     exit;
  }
  elsif (/^-i(.*)/) {
    $ext = "inc";
    $dir = "include";
    printf "Command line switch: include file\n";
  }
  elsif (/^-s(.*)/) {
    $cmd = "sdiff -w 161";
    $more = "| more";
    printf "Command line switch: side-by-side\n";
  }
  elsif (/^-b(.*)/) {
    $ext = "epx";
    $dir = "bench";
    printf "Command line switch: benchmark file\n";
  }
  elsif (/^-m(.*)/) {
    $ext = "ttx";
    $dir = "manual";
    printf "Command line switch: manual file\n";
  }
  elsif (/^-c(.*)/) {
    $opt = "-i";
    printf "Command line switch: ignore case\n";
  }
  elsif (/^-l(.*)/) {
    $ext = "listing";
    $dir = "bench";
    printf "Command line switch: listing file\n";
  }
  elsif (/^-p(.*)/) {
    $ext = "ps";
    $dir = "bench";
    printf "Command line switch: postscript file\n";
  }
  elsif (/^-u(.*)/) {
    $ext = "pl";
    $dir = "util";
    printf "Command line switch: utility (Perl) file\n";
  }
  elsif (/^-v(.*)/) {
    $ext = "vld";
    $dir = "validate";
    printf "Command line switch: validation file\n";
  }
  elsif (/^-w(.*)/) {
    $optw = "-w";
    printf "Command line switch: ignore all white space\n";
  }
  elsif (/^-d$/) {
    $_ = shift;
    $refdir = $_;
    printf "Command line switch: reference directory:$refdir\n";
  }
  elsif (/^-a(.*)/) {
    $ext = "avi";
    $dir = "bench";
    printf "Command line switch: animation (avi) file\n";
  }
  elsif (/^-g(.*)/) {
    $ext = "bmp";
```

```
    $dir = "bench";
    printf "Command line switch: graphics (bmp) file\n";
  }
  elsif (/^-f$/) {
    $optlist = "1";
    printf "Command line switch: Only the list is printed\n";
  }
  else {
    die "ERROR: unknown switch: $_\n";
  }
}
#-------------------------------------------------------------------------------
# Check if the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  die "The EUROPLEXUS environment variable is undefined!\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  die "The EUROPLEXUS directory: $epx does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# All remaining arguments are file names: do the globbing
#
@FileNames = glob (join ($",@ARGV));
#-------------------------------------------------------------------------------
# Loop on file(s) to be compared
#
foreach $name (@FileNames) {
#-------------------------------------------------------------------------------
# Set up file name
#
    $base = $name;
    $base =~ s/\.$ext$//;                       # Remove extension if present
    $extu = $ext;                               # Treat also uppercase ext!
    $extu =~ tr /[a-z]/[A-Z]/;
    $base =~ s/\.$extu$//;
    $file = "$base.$ext";
    if( $optlist eq "0"){
       print("\n\n#########>Comparing $file\n");
    }
#-------------------------------------------------------------------------------
# Verify that the file exists in the appropriate directory
#
    if ( ! defined $refdir ) {
      $from = "$epx\\$dir\\$file";                # Full file name
    }
    else {
      $from = "$refdir\\$file";                   # Relative file name
    }
    if ( ! -r $from ) {
      if ( ! defined $refdir ) {
        print "File $file does not exist in $epx\\$dir!\007\n";
      }
      else {
        print "File $file does not exist in $refdir!\007\n";
      }
    }
    else {
#-------------------------------------------------------------------------------
# Verify that the file exists in the local directory
#
      if ( ! -r $file ) {
        die "File $file does not exist in local directory!\007\n";
      }
#-------------------------------------------------------------------------------
# Diff file
#
      if( $optlist eq "1"){
         system("$cmd $opt $optw $from $file $more >test_diff.txt");
         if (!-z "test_diff.txt"){
            print $file."\n";
         }
      }
      else{
         system("$cmd $opt $optw $from $file $more");
      }
#
    }
#
}
exit;
```

epx_dup_bench.pl

```
# Duplicate EUROPLEXUS benchmark with new name
#
#-------------------------------------------------------------------------------
# Default values
#
#-------------------------------------------------------------------------------
# There must be at least two arguments (old file name and new file name)
#
if ($#ARGV < 1) {                           # Index of last argument must be >= 1
  $errmsg = "Usage: epx_dup_bench oldname<.epx> newname<.epx>\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#
#print "Index of last arg is: $#ARGV.\n";
$nar = $#ARGV + 1;
#print "No. of args is: $nar.\n";
if ($nar != 2) {
  $errmsg = "ERROR: too few or too many file names\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
else {
  $_ = shift;
  $nar = $nar - 1;
  $oldfile = $_;
  printf "Old file name: $oldfile\n";
  $oldbase = $oldfile;
  $oldbase =~ s/\.epx$//;                   # Remove '.epx' extension if present
  $oldfile = "$oldbase.epx";
  $oldgibi = "$oldbase.dgibi";
  $_ = shift;
  $nar = $nar - 1;
  $newfile = $_;
  printf "New file name: $newfile\n";
  $newbase = $newfile;
  $newbase =~ s/\.epx$//;                   # Remove '.epx' extension if present
  $newfile = "$newbase.epx";
  $newgibi = "$newbase.dgibi";
  $oldbaselc = $oldbase;
  $oldbaselc =~ tr /[A-Z]/[a-z]/;
  $oldbaseuc = $oldbase;
  $oldbaseuc =~ tr /[a-z]/[A-Z]/;
  $newbaselc = $newbase;
  $newbaselc =~ tr /[A-Z]/[a-z]/;
  $newbaseuc = $newbase;
  $newbaseuc =~ tr /[a-z]/[A-Z]/;
}
#-------------------------------------------------------------------------------
# Duplicate old .dgibi file if present
#
if (-r $oldgibi) {
  if (-r $newgibi) {
    print "File $newgibi already present in current directory\n";
    print "Please, rename it or delete it\n";
    exit;
  }
  print "Filtering: $oldgibi to produce $newgibi\n";
  open (INPUT, "type $oldgibi |");
  open (OUTPUT, ">> $newgibi");
  while (<INPUT>) {
    s/$oldbaselc/$newbaselc/;
    s/$oldbaseuc/$newbaseuc/;
    print (OUTPUT);
  }
  close OUTPUT;
  close INPUT;
}
#-------------------------------------------------------------------------------
# Loop on epx file(s) to be filtered
#
opendir(THISDIR, ".") || die "Can't open current directory\n";
@FileNames = readdir(THISDIR);
#print "FileNames: @FileNames\n";
foreach $name (@FileNames) {
  #-----------------------------------------------------------------------------
  # Test file name for matching
  #
  #print "File: $name\n";
  $_ = $name;
  if ( m/^$oldbase\w*\.epx$/ ) {
    $base = $name;
    $base =~ s/\.epx$//;                    # Remove '.epx' extension if present
    $ofile = "$base.epx";
    $nfile = $ofile;
    $nfile =~ s/$oldbase/$newbase/;
    print "Filtering: $ofile to produce $nfile\n";
    #
    # Process .epx File
    #
    open (INPUT, "type $ofile |");
    open (OUTPUT, ">> $nfile");
    while (<INPUT>) {
      s/$oldbaselc/$newbaselc/;
      s/$oldbaseuc/$newbaseuc/;
      print (OUTPUT);
    }
    close OUTPUT;
    close INPUT;
  }
}
exit;

#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_evol_64.pl

```
# SPECIAL VERSION of epx_evol_start for the 64-bit machine:
#
#  - The QuickWin version is NOT built!
#  - The -check version is NOT built!
#  - The manuals are fabricated but are NOT put on-line
#        (this is already done by the 32-bit server)
#
#-------------------------------------------------------------------------------
# Start evolution of EUROPLEXUS files. All evolution packages present in the
# FTP directory are treated. If more than one evolution package set is
# present, they are applied in order.
#-------------------------------------------------------------------------------
# Modules needed
#
use HTTP::Date;
#-------------------------------------------------------------------------------
# Redirect STDOUT and STDERR to evol.log (Fixed path!!!!)
#
$logfile = "E:\\EUROPLEXUS\\Fromcentral\\evol.log";
open STDOUT, ">>$logfile";
open STDERR, ">>$logfile";
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Get and print current date and time
#
$time = time;                       # Machine time (seconds since epoch)
```

```
$gmttime = time2str($time);        # Universal (GMT) date and time
$loctime = time2iso($time);        # This requires function time2iso to be
                                   # exported in module HTTP\Date.pm (is not
                                   # exported by default!
print "\n\n";
print "========================================";
print "========================================\n";
print "===> EUROPLEXUS 64-bit evolution(s) at JRC started on: $gmttime\n";
print "===> Local date/time: $loctime\n";
#-------------------------------------------------------------------------------
# Initialize times
#
($user0, $system0, $cuser0, $csystem0) = times;
#-------------------------------------------------------------------------------
# Default values
#
#$ftpdir = "E:\\IIS\\FTP\\EUROPLEXUS"; # Directory receiving FTP files
$ftpdir = "E:\\EUROPLEXUS\\FTP"; # Directory receiving FTP files
#
$wwwdir = "E:\\WWW\\EUROPLEXUS"; # Directory where to put on-line files
$wwwman = "$wwwdir\\MANUAL";     # Directory where to put on-line manual
$wwwexe = "$wwwdir\\EXE";        # Directory where to put on-line executable
#
#fc 02 May 2005
$copyux = "cp -p -f";
$copy = "xcopy /Q /R /Y /I";
#fc 02 May 2005
$del  = "rm -f";
#-------------------------------------------------------------------------------
$gnudir = $ENV{'GNUDIR'};
if ( ! defined $gnudir) {
  die "The GNUDIR environment variable is undefined!\007\n";
}
$rmdir = "$gnudir\\rmdir.exe";   # Full path needed! Else uses MS-DOS's rmdir!
$errors = "E:\\EUROPLEXUS\\Fromcentral\\errors.txt";
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 0 or 1 or 2
#
if ($#ARGV > 1) { # Index of last argument must be = -1 or 0 or 1
  die "Usage: epx_evol_start [-p] [-fullcmp]\007\n";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#       -p       patch (do not get evolution files by ftp)
#       -fullcmp full compilation (recompile all sources)
#
#print "Index of last arg is: $#ARGV.\n";
$nar = $#ARGV + 1;
#print "No. of args is: $nar.\n";
if ($#ARGV >= 0) { # There is at least one argument
  while ($ARGV[0] =~ /^-/) {
    $_ = shift;
    $nar = $nar - 1;
    if (/^-p(.*)/) {
      $patch = "yes";
      print "Command line switch: patch\n";
    }
    elsif (/^-fullcmp(.*)/) {
      $full_compil = "yes";
      print "Command line switch: fullcmp\n";
    }
    else {
      die "ERROR: unknown switch: $_\n";
    }
  }
}
#print "Index of last arg is: $#ARGV.\n";
#print "No. of args is: $nar.\n";
#-------------------------------------------------------------------------------
# Get any evolution files by ftp and copy them to the local directory ($ftpdir)
#
$ftplogfile = "E:\\EUROPLEXUS\\Fromcentral\\evol_getfiles.log";
$exelogfile = "E:\\EUROPLEXUS\\Fromcentral\\evol_putexe.log";
$manlogfile = "E:\\EUROPLEXUS\\Fromcentral\\evol_putman.log";
#
if (! defined $patch) {
  print "\n===> Getting any evolution files from central mirror site.\n";
  print "     (see evol_getfiles.log)\n";
  system ("epx_ftp_getfiles");
}
else {
  print "\n===> Not getting any evol. files (-p option).\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  die "The EUROPLEXUS environment variable is undefined!\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  die "The EUROPLEXUS directory: $epx does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Check existence of evolution version file, and NON-existence of
# evolution lock file
#
$evolck = "$epx\\evolution.lck";           # Evolution lock file (blocks
                                           # successive evols in case of errs)
$evonumfile = "$epx\\VERSION.txt";         # Evolution version file
if (! -r $evonumfile) {
  die "The evolution number file $evonumfile does not exist!\007\n";
}
if (-r $evolck) {
  die "Evolution is stopped because the lock file $evolck exists!\007\n";
}
#-------------------------------------------------------------------------------
# Set and check existence of auxiliary directories
#
$bkpdir = "$epx\\Backup";                  # Directory holding backup info
$evodir = "$epx\\Fromcentral";             # Directory used for evolution
if ( ! -d $bkpdir) {
  die "The EUROPLEXUS directory: $bkpdir does not exist!\007\n";
}
if ( ! -d $evodir) {
  die "The EUROPLEXUS directory: $evodir does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Make sure we are in $evodir (so we may use . instead of $evodir)
#
$curdir = `pwd`;
chop $curdir;
if ( $curdir ne $evodir ) {
  print "\nChanging directory to $evodir.\n";
  chdir "$evodir" ||
  die "Can't chdir to $evodir!\007\n";
}
#-------------------------------------------------------------------------------
# Set and check existence of target directories
#
$srcdir = "$epx\\source";
$incdir = "$epx\\include";
$libdir = "$epx\\library";
$mandir = "$epx\\manual";
$manfildir = "$epx\\manual_filtered";
$hacdir = "$epx\\manual\\hacha";
$bendir = "$epx\\bench";
$hisdir = "$epx\\history";
$exedir = "$epx\\exe";
$trcdir = "$epx\\trace";
$moddir = "$epx\\module";
$MPImoddir = "$epx\\module_mpi";
$valdir = "$epx\\validate";
#
if ( ! -d $srcdir) {
  die "The EUROPLEXUS directory: $srcdir does not exist!\007\n";
}
if ( ! -d $incdir) {
  die "The EUROPLEXUS directory: $incdir does not exist!\007\n";
}
if ( ! -d $libdir) {
  die "The EUROPLEXUS directory: $libdir does not exist!\007\n";
}
if ( ! -d $mandir) {
  die "The EUROPLEXUS directory: $mandir does not exist!\007\n";
}
if ( ! -d $manfildir) {
  die "The EUROPLEXUS directory: $manfildir does not exist!\007\n";
}
if ( ! -d $hacdir) {
  die "The EUROPLEXUS directory: $hacdir does not exist!\007\n";
}
if ( ! -d $bendir) {
  die "The EUROPLEXUS directory: $bendir does not exist!\007\n";
}
if ( ! -d $hisdir) {
  die "The EUROPLEXUS directory: $hisdir does not exist!\007\n";
}
if ( ! -d $exedir) {
  die "The EUROPLEXUS directory: $exedir does not exist!\007\n";
}
if ( ! -d $trcdir) {
  die "The EUROPLEXUS directory: $trcdir does not exist!\007\n";
}
if ( ! -d $moddir) {
  die "The EUROPLEXUS directory: $moddir does not exist!\007\n";
}
if ( ! -d $valdir) {
  die "The EUROPLEXUS directory: $valdir does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Set and check existence of objects library
#
$objlib = "$libdir\\libplex\.lib";
if ( ! -r $objlib) {
  die "The EUROPLEXUS objects library: $objlib does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Set and check existence of the standard executable
#
$stdexe = "$exedir\\europlexus\.exe";
if ( ! -r $stdexe) {
  die "The EUROPLEXUS standard executable: $stdexe does not exist!\007\n";
}
# Set also the QuickWin executable
$stdqwexe = "$exedir\\europlexusqw\.exe";
# Set also the MPI executable
$stdMPIexe = "$exedir\\europlexus_mpi\.exe";
#-------------------------------------------------------------------------------
# Read the number of the previous evolution
#
$evonum = `cat $evonumfile`;
chop $evonum;
print "\n===> The current evolution index is  : \# $evonum\n";
$numok = $evonum + 1;
print "===> The next evolution index must be: \# $numok\n";
#-------------------------------------------------------------------------------
# Treat all the evolution "resume" files, i.e., files of the
# form R_nnnnxjjmmmyy.txt in $ftpdir. These contain the list of
# evolution packages relative to each evolution, i.e names of files of the
# form [ISBMV]_NNNNxDDMMMYY.tar.gz that must also be in $ftpdir.
#
opendir(DIR, $ftpdir) || die "Can't open $ftpdir\n";
@FileNames = sort readdir(DIR);
closedir(DIR);
#
foreach $file (@FileNames) {
  $_ = $file;
  $dirf = "$ftpdir\\$file";
  if ( -r $dirf ) {
#-------------------------------------------------------------------------------
    # NOTE: the preceding test on file existence is necessary, because
    # treatment of a file in the list involves treatment (and deletion)
    # of all its matching files, if any!
    #
    if ( m/^R_\d\d\d\dx\d\d[A-Z][a-z][a-z]\d\d.txt$/ ) {
      print "\n===> Evolution resume file found: $file\n";
#-------------------------------------------------------------------------------
      # Evolution resume file found: open it and read names of
      # the associated evolution packages (I_, S_, B_, M_, V_), of which
      # zero to five must be listed, and which must be present in the
      # FTP directory.
      #
```

```
      # The case zero corresponds to an evolution which is empty because
      # it failed at the central mirror site. In this case the trace is
      # sent anyway (it will contain the error message). The only effects
      # on the mirror site are in this case:
      #   - the evolution counter is incremented;
      #   - the resume (R_) and trace (T_) files are moved to $trcdir
      #
      $rfile = $file;
      $rbase = $rfile; $rbase =~ s/\.txt$//;
      #
      $numdate = $rbase; $numdate =~ s/^R_//;     # Number & date (NNNNxDDMMMYY)
      $date = $numdate; $date =~ s/^.....//;      # Date (DDMMMYY)
      $num = $numdate; $num =~ s/x.......$//;     # Number (with leading 0s)
      $lognum = $num;
      $num =~ s/^0*//;                            # Number (no leading 0s)
      print "The evolution index is: $num\n";
      #
#-------------------------------------------------------------------------------
      # Is the index of this evolution the right one?
      #
      if ($num == $numok) {
        #
        $evolog = "$evodir\\$lognum\.log";
        #
        print "\n===> The evolution index is OK: the evolution starts.\n";
        print "\n  Redirecting STDOUT and STDERR to $evolog\n     ...\n";
#-------------------------------------------------------------------------------
        # Redirect STDOUT and STDERR to this evolution's log file ($evolog)
        #
        close STDOUT;
        close STDERR;
        open STDOUT, ">>$evolog";
        open STDERR, ">>$evolog";
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
        $oldfh = select(STDOUT); $| = 1; select($oldfh);
        $oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Get and print current date and time
#
        $time = time;
        $gmttime = time2str($time);
        $loctime = time2iso($time);
        # add 8 initial dummy lines
        # to avoid that possible error messages
        # may overwrite first part of log file
        print "============================================================\n";
        print "============================================================\n";
        print "============================================================\n";
        print "============================================================\n";
        print "============================================================\n";
        print "============================================================\n";
        print "============================================================\n";
        print "============================================================\n";
        print "===> JRC EUROPLEXUS 64-bit evolution \# $lognum started on: $gmtt
ime\n";
        print "===> Local date/time: $loctime\n";
#-------------------------------------------------------------------------------
# Initialize times
#
        ($usera, $systema, $cusera, $csystema) = times;
#-------------------------------------------------------------------------------
        # Set the lock file: it will be reset only when the evolution
        # is successfully terminated
        #
        system("echo $num >$evolck");
        print "\n===> The lock file has been set\n\n";
        #
        print "This evolution should contain the following packages:\n";
        print "  Trace file (mandatory)\n";
        #
        $dirr = "$ftpdir\\$rfile";
        #
        open (RESUME, $dirr);
        while (<RESUME>) {
          chop;
          if ( m/^[ISBMV]_\d\d\d\dx\d\d[A-Z][a-z][a-z]\d\d.tar.gz$/ ) {
            $entry = $_;
            $entry =~ s/^._.....//;
            $entry =~ s/\.tar\.gz$//;
            if ($entry eq $date) {
              #
              # The entry in the resume file matches the evolution date
              #
              if    ( m/^I/ ) {$ifile = $_; print "  Includes package.\n"}
              elsif ( m/^S/ ) {$sfile = $_; print "  Sources package.\n"}
              elsif ( m/^B/ ) {$bfile = $_; print "  Benchmarks package.\n"}
              elsif ( m/^M/ ) {$mfile = $_; print "  Manuals package.\n"}
              elsif ( m/^V/ ) {$vfile = $_; print "  Validations package.\n"}
              else {
                $errmsg =
                "The resume file contains an ambiguous name: $_!\007\n";
                print "$errmsg";
                &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
              }
            }
          }
        }
        close RESUME;
#-------------------------------------------------------------------------------
        # Verify that the trace file corresponding to the resume file
        # is present in $ftpdir
        #
        print "It contains:\n";
        $tfile = $rfile; $tfile =~ s/^R/T/;
        $dirt = "$ftpdir\\$tfile";
        #
        if (! -r $dirt) {
          $errmsg = "The trace file $tfile missing in $ftpdir!\007\n";
          print "$errmsg";
          &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
        }
        else {
          print "  Trace file $dirt present.\n";
        }
#-------------------------------------------------------------------------------
        # Copy the existing packages (I_, S_, B_, M_, V_), if any,
        # to the evolution directory.
        #
        if (defined($ifile)) {
          $diri = "$ftpdir\\$ifile";
          if ( -r $diri ) {
            print "  Includes package $diri present.\n";
            system("$copy $diri $evodir");
          }
          else {
            $errmsg = "File $ifile listed in $rfile missing in $ftpdir!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#-------------------------------------------------------------------------------
        if (defined($sfile)) {
          $dirs = "$ftpdir\\$sfile";
          if ( -r $dirs ) {
            print "  Sources package $dirs present.\n";
            system("$copy $dirs $evodir");
          }
          else {
            $errmsg = "File $sfile listed in $rfile missing in $ftpdir!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#-------------------------------------------------------------------------------
        if (defined($mfile)) {
          $dirm = "$ftpdir\\$mfile";
          if ( -r $dirm ) {
            print "  Manuals package $dirm present.\n";
            system("$copy $dirm $evodir");
          }
          else {
            $errmsg = "File $mfile listed in $rfile missing in $ftpdir!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#-------------------------------------------------------------------------------
        if (defined($bfile)) {
          $dirb = "$ftpdir\\$bfile";
          if ( -r $dirb ) {
            print "  Benchmarks package $dirb present\n";
            system("$copy $dirb $evodir");
          }
          else {
            $errmsg = "File $bfile listed in $rfile missing in $ftpdir!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#-------------------------------------------------------------------------------
        if (defined($vfile)) {
          $dirv = "$ftpdir\\$vfile";
          if ( -r $dirv ) {
            print "  Validations package $dirv present\n";
            system("$copy $dirv $evodir");
          }
          else {
            $errmsg = "File $vfile listed in $rfile missing in $ftpdir!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#-------------------------------------------------------------------------------
#fc      # Check whether all sources have to be recompiled:
#fc      # if there is a sources package and it contains at least one module
#fc      #
#fc      if (defined ($sfile)) {
#fc        print "\n===> Checking source package for full recompilation\n";
#fc        print "===> Gunzipping and untarring $sfile\n";
#fc        system("gunzip $sfile");
#fc        $sbase = $sfile; $sbase =~ s/\.tar\.gz$//;
#fc        system("tar xvf $sbase.tar 2>>$errors");
#fc        system("gzip -f -9 $sbase.tar");
#fc        #
#fc        # Build up list of files in local directory
#fc        opendir (LOCALDIR, ".");
#fc        @files = readdir (LOCALDIR);
#fc        closedir (LOCALDIR);
#fc        # Check whether there are module files (m_*.ff)
#fc        foreach $file (@files) {
#fc          $_ = $file;
#fc          if ( m/^m_[a-z0-9_]+\.ff/ ) {
#fc            $full_compil = "yes";
#fc          }
#fc        }
#fc      }
#-------------------------------------------------------------------------------
        # Process the includes package, if any
        #
        if (defined ($ifile)) {
          print "\n===> Testing includes\n";
          print "===> Gunzipping and untarring $ifile\n";
          system("gunzip $ifile");
          $ibase = $ifile; $ibase =~ s/\.tar\.gz$//;
          system("tar xvf $ibase.tar 2>>$errors");
          system("gzip -f -9 $ibase.tar");
#-------------------------------------------------------------------------------
          # Retrieve the includers, but not if full recompilation is needed
          #
          if ( ! defined ($full_compil)) {
            unlink "epx_get_includers.err";
            system("epx_get_includers");
            if ( -r "epx_get_includers.err" ) {
              $errmsg = "Die epx_evol_start: ERROR in epx_get_includers!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
        }
#-------------------------------------------------------------------------------
        # Process the sources package, if any
        #
        if (defined ($sfile)) {
          print "\n===> Testing sources\n";
```

```
          #
          # If full recompilation needed, retrieve all sources
          if (defined ($full_compil)) {
            print "\n===> Retrieving all sources (full recompilation needed)\n";
            if (system ("$copy $srcdir\\*.ff .")) {
              $errmsg = "Die epx_evol_start: ERROR retrieve all sources!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          #
          print "===> Gunzipping and untarring $sfile\n";
          system("gunzip $sfile");
          $sbase = $sfile; $sbase =~ s/\.tar\.gz$//;
          system("tar xvf $sbase.tar 2>>$errors");
          system("gzip -f -9 $sbase.tar");
#-------------------------------------------------------------------------------
          # Verify that a proper history file is contained within the sources
          # package
          #
          $sfileh = $sfile;
          $sfileh =~ s/\.tar\.gz$//;
          $sfileh =~ s/^S_/Cs/;
          $sfileh = "$sfileh\.hist";
          if (! -r $sfileh) {
            $errmsg = "History file $sfileh missing in sources package!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
#-------------------------------------------------------------------------------
          # Remove the epx_ordo.txt file if present, because it
          # contains only the names in the sources package and NOT those
          # of the retrieved includers.
          # A new, complete epx_ordo.txt will be automatically
          # generated when invoking epx_cmp without file names!
          # Same thing in case full recompilation is needed!
          #
          if (defined ($ifile) || defined ($full_compil)) {
            if ( -r "epx_ordo.txt" ) {
              print "\n===> Removing epx_ordo.txt file from sources package.\n";
              unlink "epx_ordo.txt";
            }
          }
#-------------------------------------------------------------------------------
          # Compile the sources
          #
          unlink "epx_cmp.err";

          system("epx_cmp -q -o");

          if ( -r "epx_cmp.err" ) {
            $errmsg = "Die epx_evol_start: compilation ERROR (epx_cmp)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#-------------------------------------------------------------------------------
        # Link and produce an executable module, if necessary. If there are
        # benchmarks but neither sources nor includes in evolution, then
        # no link is performed, but the standard executable is
        # copied locally under the name epx.exe.
        #
        if (defined($sfile) || defined($ifile) || defined($bfile)) {
          #
          # Benchs are necessary because this is not a "manuals-only"
          # evolution!
          #
          if (defined($sfile) || defined($ifile)) {
            print "\n===> Linking to produce a local library (libplex.lib)\n";
            print "\n===> and executable (epx.exe).\n";
            #
            unlink "epx_lk.err";
            if (defined ($full_compil)) {
              # Full recompilation has been done. Build up the local library
              # ex-novo, i.e. by ignoring the standard one (-L switch)
              system("epx_lk -L -o");
            }
            else {
              system("epx_lk -l -o");
            }
            if ( -r "epx_lk.err" ) {
              $errmsg = "Die epx_evol_start: ERROR in epx_lk!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          else {
            print "\n===> Copying standard executable to epx.exe\n";
#fc use copyux otherwise it prompts user: is it a file or directory name!
            system ("$copyux $stdexe epx.exe");
          }
#-------------------------------------------------------------------------------
          # Execute the benchmark test suite
          #
          print "\n===> Executing the benchmark test suite\n";
          #
          # Process the benchmarks package, if any
          #
          if (defined ($bfile)) {
            print "===> Gunzipping and untarring $bfile\n";
            system("gunzip $bfile");
            $bbase = $bfile; $bbase =~ s/\.tar\.gz$//;
            system("tar xvf $bbase.tar 2>>$errors");
            system("gzip -f -9 $bbase.tar");
            #
            # Verify that a proper history file is contained within the
            # benchmarks package
            #
            $bfileh = $bfile;
            $bfileh =~ s/\.tar\.gz$//;
            $bfileh =~ s/^B_/Cb/;
            $bfileh = "$bfileh\.hist";
            if (! -r $bfileh) {
              $errmsg =
              "History file $bfileh missing in benchmarks package!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          #
          # Run the local tests (if any). Run also the standard
          # benchmark tests, if either the sources or the includes
          # (or both) have changed.
          #
          if (defined($sfile) || defined($ifile)) {
            $local = "";
          }
          else {
            $local = "-l";
          }
          #
          unlink "epx_test_benchmarks.err";
          system("epx_test_benchmarks $local");
          if ( -r "epx_test_benchmarks.err" ) {
            $errmsg = "Die epx_evol_start: ERROR in epx_test_benchmarks!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#-------------------------------------------------------------------------------
        # Update the includes library if necessary
        #
        if (defined($ifile)) {
          print "\n===> Updating the includes library\n";
          if (system ("$copy *.inc $incdir")) {
            $errmsg = "Die epx_evol_start: ERROR updating the includes!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#-------------------------------------------------------------------------------
        # Update the sources library if necessary
        #
        if (defined($sfile)) {
          print "\n===> Updating the sources library\n";
          if (system ("$copy *.ff $srcdir")) {
            $errmsg = "Die epx_evol_start: ERROR updating the sources!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#-------------------------------------------------------------------------------
        # Update the modules (.mod) library if necessary
        #
        if (defined($ifile) || defined($sfile)) {
          #
          # Verify that there are > 0 mod files, to avoid copy error
          #
          $nummod = `ls -l *.mod 2>>$errors | wc -l`;
          chop $nummod;
          $nummod =~ s/[ \t]//g;
          if ($nummod > 0) {
            print "\n===> Updating the modules library\n";
            if (system ("$copy *.mod $moddir")) {
              $errmsg = "Die epx_evol_start: ERROR updating the modules!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
        }
#-------------------------------------------------------------------------------
        # Update the objects (.lib) library if necessary
        #
        if (defined($ifile) || defined($sfile)) {
          print "\n===> Updating the objects library\n";
          if (system ("$copy libplex.lib $libdir")) {
            $errmsg = "Die epx_evol_start: ERROR updating the library!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#-------------------------------------------------------------------------------
        # Update the executable if necessary
        #
        if (defined($sfile) || defined($ifile)) {
          print "\n===> Updating the executable\n";
          if (system ("$copy epx.exe $stdexe")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the executable!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          #
          # Update also the on-line executable
          #
          print "\n===> Update the on-line executable (see evol_putexe.log)\n";
#fc use copyux because of LOCAL copy!
          system ("$copyux epx.exe europlexus64.exe");
          system ("gzip -f -9 europlexus64.exe");
#         if (system ("$copy europlexus64.exe.gz $wwwexe\\europlexus64\.exe\.gz"
)) {
#           $errmsg =
#           "Die epx_evol_start: ERROR updating the on-line executable!\007\n";
#           print "$errmsg";
#           &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
#         }
          system ("epx_ftp_putexe_64");
#
          system ("$del europlexus64.exe");
          #-----------------------------------------------------------------
          # Generate also the QuickWin executable and update it
#         print "\n===> Linking to produce a local library (libplex.lib)\n";
#         print "\n===> and QuickWin executable (epxqw.exe).\n";
          #
#         unlink "epx_lk.err";
#         if (defined ($full_compil)) {
            # Full recompilation has been done. Build up the local library
            # ex-novo, i.e. by ignoring the standard one (-L switch)
#           system("epx_lk -L -o -w");
#         }
#         else {
#           system("epx_lk -l -o -w");
#         }
#         if ( -r "epx_lk.err" ) {
#           $errmsg = "Die epx_evol_start: ERROR in epx_lk -w!\007\n";
#           print "$errmsg";
```

```
#           &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
#         }
#         print "\n===> Updating the QuickWin executable\n";
#         if (system ("$copy epxqw.exe $stdqwexe")) {
#           $errmsg =
#           "Die epx_evol_start: ERROR updating the QuickWin executable!\007\n";
#           print "$errmsg";
#           &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
#         }
          # Update also the QuickWin on-line executable
#         print "\n===> Update the QuickWin on-line executable\n";
#         print " (see evol_putexe.log)\n";
#fc use copyux because of LOCAL copy!
#         system ("$copyux epxqw.exe europlexusqw.exe");
#         system ("gzip -f -9 europlexusqw.exe");
#         system ("epx_ftp_putexe -w");
#         system ("$del europlexusqw.exe");
          #-------------------------------------------------------------------
          #-------------------------------------------------------------------
          # Generate also the MPI executable and update it
          #
          print "\n===> Updating the MPI executable\n";
          #
          # Remove work directory MPI if already existing
          if (-d "MPI") {
            print "\n  ===> Deleting the previous MPI directory\n";
# use rd instead of rmdir since in some cases rmdir seems to use
# GNU's rmdir instead of DOS rmdir (which is the one I want to use here)
#           system ("rmdir /s/q MPI >>$errors 2>>&1");
            system ("rd /s/q MPI >>$errors 2>>&1");
#
          }
          # Create a work directory MPI and go in it
          print "\n  ===> Creating the MPI directory\n";
          if (system ("mkdir MPI")) {
            $errmsg =
            "Die epx_evol_start: ERROR mkdir MPI!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "\n  ===> Changing to the MPI directory\n";
          if (system ("chdir MPI")) {
            $errmsg =
            "Die epx_evol_start: ERROR chdir MPI!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          if (system ("$copy ..\\*.ff .")) {
            $errmsg =
            "Die epx_evol_start: ERROR copying sources for MPI!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          if (defined($ifile)) {
            if (system ("$copy ..\\*.inc .")) {
              $errmsg =
              "Die epx_evol_start: ERROR copying includes for MPI!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          print "\n===> Compiling sources for MPI version\n";
          system("epx_cmp -q -o -M");
          if ( -r "epx_cmp.err" ) {
            $errmsg = "Die epx_evol_start: MPI compilation ERROR (epx_cmp)!\007\
n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "\n===> Linking MPI version\n";
          if (defined ($full_compil)) {
            # Full recompilation has been done. Build up the local library
            # ex-novo, i.e. by ignoring the standard one (-L switch)
            system("epx_lk -L -o -M");
          }
          else {
            system("epx_lk -l -o -M");
          }
          if ( -r "epx_lk.err" ) {
            $errmsg = "Die epx_evol_start: MPI linkage ERROR in epx_lk!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "\n===> Updating the MPI executable\n";
          if (system ("$copy epx.exe $stdMPIexe")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the MPI executable!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          # Update also the MPI on-line executable
          print "\n===> Update the MPI on-line executable\n";
          print " (see evol_putexe.log)\n";
#fc use copyux because of LOCAL copy!
          system ("$copyux epx.exe europlexus64_mpi.exe");
          system ("gzip -f -9 europlexus64_mpi.exe");
          system ("epx_ftp_putexe_64 -M");
          system ("$del europlexus64_mpi.exe");
          # Update the modules (.mod) library if necessary
          #
          #
          # Verify that there are > 0 mod files, to avoid copy error
          #
          $nummod = `ls -l *.mod 2>>$errors | wc -l`;
          chop $nummod;
          $nummod =~ s/[ \t]//g;
          if ($nummod > 0) {
            print "\n===> Updating the MPI modules library\n";
            if (system ("$copy *.mod $MPImoddir")) {
              $errmsg = "Die epx_evol_start: ERROR updating the MPI modules!\007
\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          # Update the objects (.lib) library if necessary
          #
          print "\n===> Updating the MPI objects library\n";
          if (system ("$copy libplex_MPI.lib $libdir")) {
            $errmsg = "Die epx_evol_start: ERROR updating the library!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          # Return to parent directory
          print "\n  ===> Changing back to parent directory\n";
          if (system ("chdir ..")) {
            $errmsg =
            "Die epx_evol_start: ERROR chdir ..!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          # Clean up (remove entire temporary directory MPI)
          print "\n  ===> Removing the MPI work directory\n";
          system ("rd /s/q MPI >>$errors 2>>&1");
          #
        }
#-------------------------------------------------------------------------------
          # Update the benchmarks library if necessary
          #
          if (defined($sfile) || defined($ifile) || defined($bfile)) {
            print "\n===> Updating the benchmarks\n";
            #
            # Input data file is mandatory
            #
            if (system ("$copy *.epx $bendir")) {
              $errmsg =
              "Die epx_evol_start: ERROR updating the benchmarks(1)!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            #
            # Mesh file is optional
            # Verify that there are > 0 msh files, to avoid copy error
            #
            $nummsh = `ls -l *.msh 2>>$errors | wc -l`;
            chop $nummsh;
            $nummsh =~ s/[ \t]//g;
            if ($nummsh > 0) {
              if (system ("$copy *.msh $bendir")) {
                $errmsg =
                "Die epx_evol_start: ERROR updating the benchmarks(2)!\007\n";
                print "$errmsg";
                &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
              }
            }
            #
            # Zip file (bm_*.zip) is optional
            # Verify that there are > 0 zip files, to avoid copy error
            #
            $numzip = `ls -l bm_*.zip 2>>$errors | wc -l`;
            chop $numzip;
            $numzip =~ s/[ \t]//g;
            if ($numzip > 0) {
              if (system ("$copy bm_*.zip $bendir")) {
                $errmsg =
                "Die epx_evol_start: ERROR updating the benchmarks(3)!\007\n";
                print "$errmsg";
                &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
              }
            }
            #
            # Listing file is mandatory
            #
            if (system ("$copy *.listing $bendir")) {
              $errmsg =
              "Die epx_evol_start: ERROR updating the benchmarks(4)!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            #
            # PostScript file is optional
            # Verify that there are > 0 ps files, to avoid copy error
            #
            $numps = `ls -l *.ps 2>>$errors | wc -l`;
            chop $numps;
            $numps =~ s/[ \t]//g;
            if ($numps > 0) {
              if (system ("$copy *.ps $bendir")) {
                $errmsg =
                "Die epx_evol_start: ERROR updating the benchmarks(5)!\007\n";
                print "$errmsg";
                &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
              }
            }
            #
            # BMP file(s) are optional
            # Verify that there are > 0 bmp files, to avoid copy error
            #
            $numbmp = `ls -l *.bmp 2>>$errors | wc -l`;
            chop $numbmp;
            $numbmp =~ s/[ \t]//g;
            if ($numbmp > 0) {
              if (system ("$copy *.bmp $bendir")) {
                $errmsg =
                "Die epx_evol_start: ERROR updating the benchmarks(6)!\007\n";
                print "$errmsg";
                &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
              }
            }
            #
            # AVI file(s) are optional
            # Verify that there are > 0 avi files, to avoid copy error
            #
            $numavi = `ls -l *.avi 2>>$errors | wc -l`;
            chop $numavi;
            $numavi =~ s/[ \t]//g;
            if ($numavi > 0) {
              if (system ("$copy *.avi $bendir")) {
                $errmsg =
                "Die epx_evol_start: ERROR updating the benchmarks(7)!\007\n";
                print "$errmsg";
                &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
              }
            }
            #
            # PVD file(s) are optional
```

```
          # Verify that there are > 0 pvd files, to avoid copy error
          #
          $numpvd = `ls -1 *.pvd 2>>$errors | wc -l`;
          chop $numpvd;
          $numpvd =~ s/[ \t]//g;
          if ($numpvd > 0) {
            if (system ("$copy *.pvd $bendir")) {
              $errmsg =
              "Die epx_evol_start: ERROR updating the benchmarks(8)!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          #
          # VTU file(s) are optional
          # Verify that there are > 0 vtu files, to avoid copy error
          #
          $numvtu = `ls -1 *.vtu 2>>$errors | wc -l`;
          chop $numvtu;
          $numvtu =~ s/[ \t]//g;
          if ($numvtu > 0) {
            if (system ("$copy *.vtu $bendir")) {
              $errmsg =
              "Die epx_evol_start: ERROR updating the benchmarks(9)!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
        }
#-------------------------------------------------------------------------------
        # Process the manuals package, if any
        #
        if (defined($mfile)) {
          print "\n===> Testing manuals\n";
          print "===> Gunzipping and untarring $mfile\n";
          system("gunzip $mfile");
          $mbase = $mfile; $mbase =~ s/\.tar\.gz$//;
          system("tar xvf $mbase.tar 2>>$errors");
          system("gzip -f -9 $mbase.tar");
          #
          # Verify that a proper history file is contained within the
          # manuals package
          #
          $mfileh = $mfile;
          $mfileh =~ s/\.tar\.gz$//;
          $mfileh =~ s/^M_/Cm/;
          $mfileh = "$mfileh\.hist";
          if (! -r $mfileh) {
            $errmsg = "History file $mfileh missing in manuals package!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
#-------------------------------------------------------------------------------
          # Test the manuals
          #
          unlink "epx_test_manuals.err";
          system("epx_test_manuals");
          if ( -r "epx_test_manuals.err" ) {
            $errmsg = "Die epx_evol_start: ERROR in epx_test_manuals!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
#-------------------------------------------------------------------------------
          # Update the manuals library
          #
          print "\n===> Updating the manuals\n";
          if (system ("$copy *.ttx $mandir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the manuals(1)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  LaTeX sources updated.\n";
#
          if (system ("$copy *.tex $manfildir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the filtered manuals(1)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  LaTeX filtered sources updated.\n";
#
          if (system ("$copy manual.dvi $mandir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the manuals(2)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  DVI version updated.\n";
#
          if (system ("$copy manual.ps $mandir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the manuals(3)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  PostScript version updated.\n";
#
          if (system ("$copy manual.pdf $mandir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the manuals(4)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  PDF version updated.\n";
#
          if (system ("$copy manual.html $mandir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the manuals(5)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  HTML (Tth) version updated.\n";
#
          if (system ("$copy manual_h.html $mandir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the manuals(6)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  HTML (Hevea/Hacha) monolithic version updated.\n";
#
          if (system ("$copy hacha\\*.* $hacdir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the manuals(7)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  HTML (Hevea/Hacha) split version updated.\n";
#
          #
          # Update also the on-line manual (PDF + Hevea/Hacha split versions)
          #
#         if (system ("$copy manual.pdf $wwwman")) {
#           $errmsg =
#           "Die epx_evol_start: ERROR updating on-line PDF manual!\007\n";
#           print "$errmsg";
#           &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
#         }
#         system ("epx_ftp_putman");
#
#         print "  PDF on-line version update (see evol_putman.log).\n";
#
#         if (system ("$copy hacha\\*.* $wwwman")) {
#           $errmsg =
#           "Die epx_evol_start: ERROR updating on-line HTML manual!\007\n";
#           print "$errmsg";
#           &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
#         }
#         print "  HTML(Hacha) on-line version update (see evol_putman.log).\n";
        }
#-------------------------------------------------------------------------------
        # Process the validations package, if any
        #
        if (defined($vfile)) {
          print "\n===> Testing validations\n";
          print "===> Gunzipping and untarring $vfile\n";
          system("gunzip $vfile");
          $vbase = $vfile; $vbase =~ s/\.tar\.gz$//;
          system("tar xvf $vbase.tar 2>>$errors");
          system("gzip -f -9 $vbase.tar");
          #
          # Verify that a proper history file is contained within the
          # validations package
          #
          $vfileh = $vfile;
          $vfileh =~ s/\.tar\.gz$//;
          $vfileh =~ s/^V_/Cv/;
          $vfileh = "$vfileh\.hist";
          if (! -r $vfileh) {
            $errmsg = "History file $vfileh missing in validat. package!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
#-------------------------------------------------------------------------------
          # Update the validations library
          #
          print "\n===> Updating the validations\n";
          if (system ("$copy vl_*.vld $valdir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the validations(1)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  Validation description files (vl_*.vld) updated.\n";
#
          if (system ("$copy vl_*.zip $valdir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the validations(2)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  Validation package files (vl_*.zip) updated.\n";
        }
#-------------------------------------------------------------------------------
        # Process the history file(s), if any, and update the histories
        #
        if (defined($sfileh) || defined($bfileh) || defined($mfileh) ||
            defined($vfileh)) {
          print "\n===> Processing the history file(s)\n";
          unlink "epx_evol_histories.err";
          if (defined($sfileh)) {
            print "  Processing the sources/includes history file(s)\n";
            system("epx_evol_histories $sfileh");
            if ( -r "epx_evol_histories.err" ) {
              $errmsg =
              "Die epx_evol_start: ERROR in epx_evol_histories!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          if (defined($bfileh)) {
            print "  Processing the benchmarks history file(s)\n";
            system("epx_evol_histories $bfileh");
            if ( -r "epx_evol_histories.err" ) {
              $errmsg =
              "Die epx_evol_start: ERROR in epx_evol_histories!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          if (defined($mfileh)) {
            print "  Processing the manuals history file(s)\n";
            system("epx_evol_histories $mfileh");
            if ( -r "epx_evol_histories.err" ) {
              $errmsg =
              "Die epx_evol_start: ERROR in epx_evol_histories!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          if (defined($vfileh)) {
            print "  Processing the validations history file(s)\n";
            system("epx_evol_histories $vfileh");
            if ( -r "epx_evol_histories.err" ) {
              $errmsg =
```

```
            "Die epx_evol_start: ERROR in epx_evol_histories!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
        print "Histories updating has successfully terminated.\n";
      }
#-------------------------------------------------------------------------------
      # Copy the resume file, the trace file, and the evolution package
      # files (if any) to the backup directory, then delete them from
      # the FTP directory.
      #
      system("$copy $dirr $bkpdir"); unlink $dirr;
      system("$copy $dirt $bkpdir"); unlink $dirt;
      if (defined($ifile)) {system("$copy $diri $bkpdir"); unlink $diri};
      if (defined($sfile)) {system("$copy $dirs $bkpdir"); unlink $dirs};
      if (defined($bfile)) {system("$copy $dirb $bkpdir"); unlink $dirb};
      if (defined($mfile)) {system("$copy $dirm $bkpdir"); unlink $dirm};
      if (defined($vfile)) {system("$copy $dirv $bkpdir"); unlink $dirv};
#-------------------------------------------------------------------------------
      # Reset the variables that are tested through "defined",
      # in case another evolution has to be performed
      #
      undef $ifile; undef $sfile; undef $bfile; undef $mfile; undef $vfile;
      undef $sfileh; undef $bfileh; undef $mfileh; undef $vfileh;
      undef $ifileunpacked;
#-------------------------------------------------------------------------------
      # The evolution is terminated:
      #   - increment the evolution index;
      #   - clean up the files in $evodir (.);
      #
      print "\n===> EVOLUTION N. $num SUCCESSFUL!\n";
      print "===> Incrementing the evolution index\n";
      #
      system ("echo $num >$evonumfile");
      $evonum = `cat $evonumfile`;
      chop $evonum;
      print "\n===> The current evolution index is  : \# $evonum\n";
      #
      $numok = $evonum + 1;
      print "===> The next evolution index must be: \# $numok\n";
      #
      print "\n===> Cleaning up the evolution directory\n";
      system("$del *.gz"); system("$del *.hist");          # Packages
      system("$del *.ff"); system("$del *.f");             # Sources
      system("$del *.inc");                                # Includes
      system("$del *.obj"); system("$del *.err");          # Compilation
      system("$del *.mod");
      system("$del *.exe"); system("$del *.lib");          # Link
      system("$del *.pdb");
      system("$del *.eco"); system("$del *.std");          # Run/benchs
      system("$del *.epx"); system("$del *.msh");
      system("$del *.listing"); system("$del *.ps");
      system("$del bm_*.log"); system("$del bm_*.zip");
      system("$del bm_*.plog"); system("$del bm_*.pin");
      system("$del bm_*.lks");
      system("$del bm_listing.lst");
      system("$del *.bmp"); system("$del *.ide");
      system("$del *.sau");
      system("$del vl_*.vld"); system("$del vl_*.zip");    # Validations
      system("$del bm_*.ali");                             # Outputs
      system("$del bm_*.alt");
      system("$del bm_*.inp");
      system("$del bm_*.k20");
      system("$del bm_*.k200"); system("$del *.k2000");
      system("$del bm_*.mtv");
      system("$del *.pun");
      system("$del bm_*.puk");
      system("$del bm_*.mgr");
      system("$del bm_*.txt");
      system("$del bm_*.avi");
      system("$del bm_*.pvd");
      system("$del bm_*.vtu");
      system("$del bm_*.str");
      system("$del bm_*.savr");
      system("$del bm_*.tpl");
      system("$del bm_*.unv");
      system("$del bm_*.xpl");
      system("$del bm_*.poc");
      system("$del bm_stderr.lst");
      system("$del epx_ordo.txt");
      system("$del epx*.exp");
      system("$del *.u18");
      system("$del *.dat");
      system("$del *.inp"); system("$del *.mif");
      system("$del *.p10"); system("$del *.pov");
      system("$del fort.*"); system("$del *.k");
      system("$del *.ttx"); system("$del *.aux");          # Manuals
      system("$del *.tex");
      system("$del gbnotice.*"); system("$del manual.*");
      system("$del manual_h.*");
      system("$del epx_pass_*.log");
      system("$del tth.log");
      if (-d "hacha") {
#       system("$del hacha\\*.*");
#       system("$rmdir hacha");
        system("rmdir /s/q hacha >>$errors 2>>&1");
      }
      system("$del _*.*");                                 # Temporary
#
# Reset full compilation flag, in case of multiple evolutions !
      if (defined ($full_compil)) {
        undef $full_compil;
      }
#-------------------------------------------------------------------------------
      # LAST OPERATION: reset the evolution lock.
      #
      print "\n===> Removing the evolution lock\n";
      unlink "$evolck";
#-------------------------------------------------------------------------------
# Report user and system CPU times for this process and for its children
#
      ($userb, $systemb, $cuserb, $csystemb) = times;
      $user    = $userb    - $usera;
      $system  = $systemb  - $systema;
      $cuser   = $cuserb   - $cusera;
      $csystem = $csystemb - $csystema;
#
      print "\nThis process used the following CPU times:\n";
      print "User   : $user\n";
      print "System : $system\n";
      print "\nThe children of this process used the following CPU times:\n";
      print "Cuser  : $cuser\n";
      print "Csystem: $csystem\n\n";
#-------------------------------------------------------------------------------
# Get and print current date and time
#
      $time = time;
      $gmttime = time2str($time);
      $loctime = time2iso($time);
      print "===> JRC EUROPLEXUS evolution \# $lognum ended   on: $gmttime\n";
      print "===> Local date/time: $loctime\n";
#-------------------------------------------------------------------------------
      # Redirect STDOUT and STDERR to $logfile
      #
      close STDOUT;
      close STDERR;
      open STDOUT, ">>$logfile";
      open STDERR, ">>$logfile";
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
      $oldfh = select(STDOUT); $| = 1; select($oldfh);
      $oldfh = select(STDERR); $| = 1; select($oldfh);
#
      print "  Redirecting STDOUT and STDERR to $logfile.\n";
      print "\n===> EVOLUTION N. $num SUCCESSFUL at $loctime!\n";
#-------------------------------------------------------------------------------
      # Mail evolution log file to project partners
      #
      print "\n===> Mailing log file $evolog to partners.\n\n";
      unlink "epx_mail.err";
      $stat = "OK";
      system("epx_mail $stat $num $evolog");
      if ( -r "epx_mail.err" ) {
        die "Die epx_evol_start: ERROR in epx_mail!\007\n";
      }
#-------------------------------------------------------------------------------
      # Move evolution log file to trace directory
      #
      print "\n===> Moving evolution log file $evolog\n";
      print "      to $trcdir.\n";
      system("$copy $evolog $trcdir"); unlink $evolog;
    }
    else {
      print "The evolution index is out of sequence: skip this evolution!\n";
    }
  }
}
#-------------------------------------------------------------------------------
# Report user and system CPU times for this process and for its children
#
($user1, $system1, $cuser1, $csystem1) = times;
#
$user    = $user1    - $user0;
$system  = $system1  - $system0;
$cuser   = $cuser1   - $cuser0;
$csystem = $csystem1 - $csystem0;
#
print "\nThis process used the following CPU times:\n";
print "User   : $user\n";
print "System : $system\n";
print "\nThe children of this process used the following CPU times:\n";
print "Cuser  : $cuser\n";
print "Csystem: $csystem\n\n";
#-------------------------------------------------------------------------------
# Get and print current date and time
#
$time = time;                       # Machine time (seconds since epoch)
$gmttime = time2str($time);         # Universal (GMT) date and time
$loctime = time2iso($time);         # This requires function time2iso to be
                                    # exported in module HTTP\Date.pm (is not
                                    # exported by default!
print "===> EUROPLEXUS evolution(s) at JRC ended   on: $gmttime\n";
print "===> Local date/time: $loctime\n";
#-------------------------------------------------------------------------------
# Copy evol.log and evol_*.log files to trace directory
#
close STDOUT;
close STDERR;
system("$copy $logfile $trcdir");

#system("$copy $ftplogfile $trcdir");
#system("$copy $exelogfile $trcdir");
#system("$copy $manlogfile $trcdir");

#
exit;


#-------------------------------------------------------------------------------
sub ERRMAIL {
  local ($logfile, $num, $evolog) = @_;
#
  close STDOUT;
  close STDERR;
  open STDOUT, ">>$logfile";
  open STDERR, ">>$logfile";
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
  $oldfh = select(STDOUT); $| = 1; select($oldfh);
  $oldfh = select(STDERR); $| = 1; select($oldfh);
#
  print "\n===> Mailing log file $evolog to partners.\n\n";
  unlink "epx_mail.err";
  $stat = "FAILED!!!";
  system("epx_mail $stat $num $evolog");
  if ( -r "epx_mail.err" ) {
    die "Die epx_evol_start: ERROR in epx_mail!\007\n";
  }
}
#-------------------------------------------------------------------------------
```

epx_evol_casa.pl

```
#-------------------------------------------------------------------------------
# Start evolution of EUROPLEXUS files. All evolution packages present in the
# FTP directory are treated. If more than one evolution package set is
# present, they are applied in order.
#-------------------------------------------------------------------------------
# Modules needed
#
use HTTP::Date;
#-------------------------------------------------------------------------------
# Redirect STDOUT and STDERR to evol.log (Fixed path!!!!)
#
$logfile = "E:\\EUROPLEXUS\\Fromcentral\\evol.log";
open STDOUT, ">>$logfile";
open STDERR, ">>$logfile";
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Get and print current date and time
#
$time = time;                       # Machine time (seconds since epoch)
$gmttime = time2str($time);         # Universal (GMT) date and time
$loctime = time2iso($time);         # This requires function time2iso to be
                                    # exported in module HTTP\Date.pm (is not
                                    # exported by default!
print "\n\n";
print "=======================================";
print "=======================================\n";
print "===> EUROPLEXUS evolution(s) at JRC started on: $gmttime\n";
print "===> Local date/time: $loctime\n";
#-------------------------------------------------------------------------------
# Initialize times
#
($user0, $system0, $cuser0, $csystem0) = times;
#-------------------------------------------------------------------------------
# Default values
#
#$ftpdir = "C:\\IIS\\FTP\\EUROPLEXUS"; # Directory receiving FTP files
$ftpdir = "E:\\EUROPLEXUS\\FTP"; # Directory receiving FTP files
#
#$wwwdir = "C:\\WWW\\EUROPLEXUS"; # Directory where to put on-line files
#$wwwman = "$wwwdir\\MANUAL";     # Directory where to put on-line manual
#$wwwexe = "$wwwdir\\EXE";        # Directory where to put on-line executable
#
#fc 02 May 2005
$copyux = "cp -p -f";
$copy = "xcopy /Q /R /Y /I";
#fc 02 May 2005
$del  = "rm -f";
#-------------------------------------------------------------------------------
$gnudir = $ENV{'GNUDIR'};
if ( ! defined $gnudir) {
  die "The GNUDIR environment variable is undefined!\007\n";
}
$rmdir = "$gnudir\\rmdir.exe";   # Full path needed! Else uses MS-DOS's rmdir!
$errors = "E:\\EUROPLEXUS\\Fromcentral\\errors.txt";
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 0 or 1 or 2 or 3 or 4 or 5 or 6
#
if ($#ARGV > 4) { # Index of last argument must be = -1 or 0 or 1 or 2 or 3 or 4
 or 5
  die "Usage: epx_evol_casa [-p] [-noqw] [-nompi] [-nocheck] [-nobench] [-fullcm
p]\007\n";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#       -p       patch (do not get evolution files by ftp)
#       -noqw    do not update the QuickWin executable
#       -nompi   do not update the MPI executable
#       -nocheck do not update the -check executable
#       -nobench do not run the benchmarks (neither old nor new)
#       -fullcmp full compilation (recompile all sources)
#
#print "Index of last arg is: $#ARGV.\n";
$nar = $#ARGV + 1;
#print "No. of args is: $nar.\n";
if ($#ARGV >= 0) { # There is at least one argument
  while ($ARGV[0] =~ /^-/) {
    $_ = shift;
    $nar = $nar - 1;
    if (/^-p(.*)/) {
      $patch = "yes";
      print "Command line switch: patch\n";
    }
    elsif (/^-noqw(.*)/) {
      $noqw = "yes";
      print "Command line switch: noqw\n";
    }
    elsif (/^-nompi(.*)/) {
      $nompi = "yes";
      print "Command line switch: nompi\n";
    }
    elsif (/^-nocheck(.*)/) {
      $nocheck = "yes";
      print "Command line switch: nocheck\n";
    }
    elsif (/^-nobench(.*)/) {
      $nobench = "yes";
      print "Command line switch: nobench\n";
    }
    elsif (/^-fullcmp(.*)/) {
      $full_compil = "yes";
      print "Command line switch: fullcmp\n";
    }
    else {
      die "ERROR: unknown switch: $_\n";
    }
  }
}
#print "Index of last arg is: $#ARGV.\n";
#print "No. of args is: $nar.\n";
#-------------------------------------------------------------------------------
# Get any evolution files by ftp and copy them to the local directory ($ftpdir)
#
$ftplogfile = "E:\\EUROPLEXUS\\Fromcentral\\evol_getfiles.log";
#$exelogfile = "C:\\EUROPLEXUS\\Fromcentral\\evol_putexe.log";
#$manlogfile = "C:\\EUROPLEXUS\\Fromcentral\\evol_putman.log";
#
if (! defined $patch) {
  print "\n===> Getting any evolution files from central mirror site.\n";
  print "     (see evol_getfiles.log)\n";
  system ("epx_ftp_getfiles");
}
else {
  print "\n===> Not getting any evol. files (-p option).\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  die "The EUROPLEXUS environment variable is undefined!\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  die "The EUROPLEXUS directory: $epx does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Check existence of evolution version file, and NON-existence of
# evolution lock file
#
$evolck = "$epx\\evolution.lck";           # Evolution lock file (blocks
                                           # successive evols in case of errs)
$evonumfile = "$epx\\VERSION.txt";         # Evolution version file
if (! -r $evonumfile) {
  die "The evolution number file $evonumfile does not exist!\007\n";
}
if (-r $evolck) {
  die "Evolution is stopped because the lock file $evolck exists!\007\n";
}
#-------------------------------------------------------------------------------
# Set and check existence of auxiliary directories
#
$bkpdir = "$epx\\Backup";                  # Directory holding backup info
$evodir = "$epx\\Fromcentral";             # Directory used for evolution
if ( ! -d $bkpdir) {
  die "The EUROPLEXUS directory: $bkpdir does not exist!\007\n";
}
if ( ! -d $evodir) {
  die "The EUROPLEXUS directory: $evodir does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Make sure we are in $evodir (so we may use . instead of $evodir)
#
$curdir = `pwd`;
chop $curdir;
if ( $curdir ne $evodir ) {
  print "\nChanging directory to $evodir.\n";
  chdir "$evodir" ||
  die "Can't chdir to $evodir!\007\n";
}
#-------------------------------------------------------------------------------
# Set and check existence of target directories
#
$srcdir = "$epx\\source";
$incdir = "$epx\\include";
$libdir = "$epx\\library";
$mandir = "$epx\\manual";
$manfildir = "$epx\\manual_filtered";
$hacdir = "$epx\\manual\\hacha";
$bendir = "$epx\\bench";
$hisdir = "$epx\\history";
$exedir = "$epx\\exe";
$trcdir = "$epx\\trace";
$moddir = "$epx\\module";
$MPImoddir = "$epx\\module_mpi";
$valdir = "$epx\\validate";
#
if ( ! -d $srcdir) {
  die "The EUROPLEXUS directory: $srcdir does not exist!\007\n";
}
if ( ! -d $incdir) {
  die "The EUROPLEXUS directory: $incdir does not exist!\007\n";
}
if ( ! -d $libdir) {
  die "The EUROPLEXUS directory: $libdir does not exist!\007\n";
}
if ( ! -d $mandir) {
  die "The EUROPLEXUS directory: $mandir does not exist!\007\n";
}
if ( ! -d $manfildir) {
  die "The EUROPLEXUS directory: $manfildir does not exist!\007\n";
}
if ( ! -d $hacdir) {
  die "The EUROPLEXUS directory: $hacdir does not exist!\007\n";
}
if ( ! -d $bendir) {
  die "The EUROPLEXUS directory: $bendir does not exist!\007\n";
}
if ( ! -d $hisdir) {
  die "The EUROPLEXUS directory: $hisdir does not exist!\007\n";
}
if ( ! -d $exedir) {
  die "The EUROPLEXUS directory: $exedir does not exist!\007\n";
}
if ( ! -d $trcdir) {
  die "The EUROPLEXUS directory: $trcdir does not exist!\007\n";
}
if ( ! -d $moddir) {
  die "The EUROPLEXUS directory: $moddir does not exist!\007\n";
}
if ( ! -d $valdir) {
  die "The EUROPLEXUS directory: $valdir does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Set and check existence of objects library
#
$objlib = "$libdir\\libplex\.lib";
```

```
if ( ! -r $objlib) {
  die "The EUROPLEXUS objects library: $objlib does not exist!\007\n";
}
#------------------------------------------------------------------------------
# Set and check existence of the standard executable
#
$stdexe = "$exedir\\europlexus\.exe";
if ( ! -r $stdexe) {
  die "The EUROPLEXUS standard executable: $stdexe does not exist!\007\n";
}
# Set also the QuickWin executable
$stdqwexe = "$exedir\\europlexusqw\.exe";
# Set also the MPI executable
$stdMPIexe = "$exedir\\europlexus_mpi\.exe";
# Set also the -check executable
$stdCHECKexe = "$exedir\\europlexus_check\.exe";
#------------------------------------------------------------------------------
# Read the number of the previous evolution
#
$evonum = `cat $evonumfile`;
chop $evonum;
print "\n===> The current evolution index is  : \# $evonum\n";
$numok = $evonum + 1;
print "===> The next evolution index must be: \# $numok\n";
#------------------------------------------------------------------------------
# Treat all the evolution "resume" files, i.e., files of the
# form R_nnnnxjjmmmyy.txt in $ftpdir. These contain the list of
# evolution packages relative to each evolution, i.e names of files of the
# form [ISBMV]_NNNNxDDMMMYY.tar.gz that must also be in $ftpdir.
#
opendir(DIR, $ftpdir) || die "Can't open $ftpdir\n";
@FileNames = sort readdir(DIR);
closedir(DIR);
#
foreach $file (@FileNames) {
  $_ = $file;
  $dirf = "$ftpdir\\$file";
  if ( -r $dirf ) {
#------------------------------------------------------------------------------
    # NOTE: the preceding test on file existence is necessary, because
    # treatment of a file in the list involves treatment (and deletion)
    # of all its matching files, if any!
    #
    if ( m/^R_\d\d\d\dx\d\d[A-Z][a-z][a-z]\d\d.txt$/ ) {
      print "\n===> Evolution resume file found: $file\n";
#------------------------------------------------------------------------------
      # Evolution resume file found: open it and read names of
      # the associated evolution packages (I_, S_, B_, M_, V_), of which
      # zero to five must be listed, and which must be present in the
      # FTP directory.
      #
      # The case zero corresponds to an evolution which is empty because
      # it failed at the central mirror site. In this case the trace is
      # sent anyway (it will contain the error message). The only effects
      # on the mirror site are in this case:
      #   - the evolution counter is incremented;
      #   - the resume (R_) and trace (T_) files are moved to $trcdir
      #
      $rfile = $file;
      $rbase = $rfile; $rbase =~ s/\.txt$//;
      #
      $numdate = $rbase; $numdate =~ s/^R_//;     # Number & date (NNNNxDDMMMYY)
      $date = $numdate; $date =~ s/^.....//;      # Date (DDMMMYY)
      $num = $numdate; $num =~ s/x.......$//;     # Number (with leading 0s)
      $lognum = $num;
      $num =~ s/^0*//;                            # Number (no leading 0s)
      print "The evolution index is: $num\n";
      #
#------------------------------------------------------------------------------
      # Is the index of this evolution the right one?
      #
      if ($num == $numok) {
        #
        $evolog = "$evodir\\$lognum\.log";
        #
        print "\n===> The evolution index is OK: the evolution starts.\n";
        print "\n  Redirecting STDOUT and STDERR to $evolog\n     ...\n";
#------------------------------------------------------------------------------
        # Redirect STDOUT and STDERR to this evolution's log file ($evolog)
        #
        close STDOUT;
        close STDERR;
        open STDOUT, ">>$evolog";
        open STDERR, ">>$evolog";
#------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
        $oldfh = select(STDOUT); $| = 1; select($oldfh);
        $oldfh = select(STDERR); $| = 1; select($oldfh);
#------------------------------------------------------------------------------
# Get and print current date and time
#
        $time = time;
        $gmttime = time2str($time);
        $loctime = time2iso($time);
        # add 8 initial dummy lines
        # to avoid that possible error messages
        # may overwrite first part of log file
        print "============================================================\n";
        print "============================================================\n";
        print "============================================================\n";
        print "============================================================\n";
        print "============================================================\n";
        print "============================================================\n";
        print "============================================================\n";
        print "============================================================\n";
        print "===> JRC EUROPLEXUS evolution \# $lognum started on: $gmttime\n";
        print "===> Local date/time: $loctime\n";
#------------------------------------------------------------------------------
# Initialize times
#
        ($usera, $systema, $cusera, $csystema) = times;
#------------------------------------------------------------------------------
        # Set the lock file: it will be reset only when the evolution
        # is successfully terminated
        #
        system("echo $num >$evolck");
        print "\n===> The lock file has been set\n\n";
        #
        print "This evolution should contain the following packages:\n";
        print "  Trace file (mandatory)\n";
        #
        $dirr = "$ftpdir\\$rfile";
        #
        open (RESUME, $dirr);
        while (<RESUME>) {
          chop;
          if ( m/^[ISBMV]_\d\d\d\dx\d\d[A-Z][a-z][a-z]\d\d.tar.gz$/ ) {
            $entry = $_;
            $entry =~ s/^._...../;
            $entry =~ s/\.tar\.gz$//;
            if ($entry eq $date) {
              #
              # The entry in the resume file matches the evolution date
              #
              if    ( m/^I/ ) {$ifile = $_; print "  Includes package.\n"}
              elsif ( m/^S/ ) {$sfile = $_; print "  Sources package.\n"}
              elsif ( m/^B/ ) {$bfile = $_; print "  Benchmarks package.\n"}
              elsif ( m/^M/ ) {$mfile = $_; print "  Manuals package.\n"}
              elsif ( m/^V/ ) {$vfile = $_; print "  Validations package.\n"}
              else {
                $errmsg =
                "The resume file contains an ambiguous name: $_!\007\n";
                print "$errmsg";
                &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
              }
            }
          }
        }
        close RESUME;
#------------------------------------------------------------------------------
        # Verify that the trace file corresponding to the resume file
        # is present in $ftpdir
        #
        print "It contains:\n";
        $tfile = $rfile; $tfile =~ s/^R/T/;
        $dirt = "$ftpdir\\$tfile";
        #
        if (! -r $dirt) {
          $errmsg = "The trace file $tfile missing in $ftpdir!\007\n";
          print "$errmsg";
          &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
        }
        else {
          print "  Trace file $dirt present.\n";
        }
#------------------------------------------------------------------------------
        # Copy the existing packages (I_, S_, B_, M_, V_), if any,
        # to the evolution directory.
        #
        if (defined($ifile)) {
          $diri = "$ftpdir\\$ifile";
          if ( -r $diri ) {
            print "  Includes package $diri present.\n";
            system("$copy $diri $evodir");
          }
          else {
            $errmsg = "File $ifile listed in $rfile missing in $ftpdir!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#------------------------------------------------------------------------------
        if (defined($sfile)) {
          $dirs = "$ftpdir\\$sfile";
          if ( -r $dirs ) {
            print "  Sources package $dirs present.\n";
            system("$copy $dirs $evodir");
          }
          else {
            $errmsg = "File $sfile listed in $rfile missing in $ftpdir!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#------------------------------------------------------------------------------
        if (defined($mfile)) {
          $dirm = "$ftpdir\\$mfile";
          if ( -r $dirm ) {
            print "  Manuals package $dirm present.\n";
            system("$copy $dirm $evodir");
          }
          else {
            $errmsg = "File $mfile listed in $rfile missing in $ftpdir!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#------------------------------------------------------------------------------
        if (defined($bfile)) {
          $dirb = "$ftpdir\\$bfile";
          if ( -r $dirb ) {
            print "  Benchmarks package $dirb present\n";
            system("$copy $dirb $evodir");
          }
          else {
            $errmsg = "File $bfile listed in $rfile missing in $ftpdir!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#------------------------------------------------------------------------------
        if (defined($vfile)) {
          $dirv = "$ftpdir\\$vfile";
          if ( -r $dirv ) {
            print "  Validations package $dirv present\n";
            system("$copy $dirv $evodir");
          }
          else {
            $errmsg = "File $vfile listed in $rfile missing in $ftpdir!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#------------------------------------------------------------------------------
#fc     # Check whether all sources have to be recompiled:
```

```
#fc     # if there is a sources package and it contains at least one module
#fc     #
#fc     if (defined ($sfile)) {
#fc       print "\n===> Checking source package for full recompilation\n";
#fc       print "===> Gunzipping and untarring $sfile\n";
#fc       system("gunzip $sfile");
#fc       $sbase = $sfile; $sbase =~ s/\.tar\.gz$//;
#fc       system("tar xvf $sbase.tar 2>>$errors");
#fc       system("gzip -f -9 $sbase.tar");
#fc       #
#fc       # Build up list of files in local directory
#fc       opendir (LOCALDIR, ".");
#fc       @files = readdir (LOCALDIR);
#fc       closedir (LOCALDIR);
#fc       # Check whether there are module files (m_*.ff)
#fc       foreach $file (@files) {
#fc         $_ = $file;
#fc         if ( m/^m_[a-z0-9_]+\.ff/ ) {
#fc           $full_compil = "yes";
#fc         }
#fc       }
#fc     }
#-------------------------------------------------------------------------------
        # Process the includes package, if any
        #
        if (defined ($ifile)) {
          print "\n===> Testing includes\n";
          print "===> Gunzipping and untarring $ifile\n";
          system("gunzip $ifile");
          $ibase = $ifile; $ibase =~ s/\.tar\.gz$//;
          system("tar xvf $ibase.tar 2>>$errors");
          system("gzip -f -9 $ibase.tar");
#-------------------------------------------------------------------------------
          # Retrieve the includers, but not if full recompilation is needed
          #
          if ( ! defined ($full_compil)) {
            unlink "epx_get_includers.err";
            system("epx_get_includers");
            if ( -r "epx_get_includers.err" ) {
              $errmsg = "Die epx_evol_start: ERROR in epx_get_includers!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
        }
#-------------------------------------------------------------------------------
        # Process the sources package, if any
        #
        if (defined ($sfile)) {
          print "\n===> Testing sources\n";
          #
          # If full recompilation needed, retrieve all sources
          if (defined ($full_compil)) {
            print "\n===> Retrieving all sources (full recompilation needed)\n";
            if (system ("$copy $srcdir\\*.ff .")) {
              $errmsg = "Die epx_evol_start: ERROR retrieve all sources!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          #
          print "===> Gunzipping and untarring $sfile\n";
          system("gunzip $sfile");
          $sbase = $sfile; $sbase =~ s/\.tar\.gz$//;
          system("tar xvf $sbase.tar 2>>$errors");
          system("gzip -f -9 $sbase.tar");
#-------------------------------------------------------------------------------
          # Verify that a proper history file is contained within the sources
          # package
          #
          $sfileh = $sfile;
          $sfileh =~ s/\.tar\.gz$//;
          $sfileh =~ s/^S_/Cs/;
          $sfileh = "$sfileh\.hist";
          if (! -r $sfileh) {
            $errmsg = "History file $sfileh missing in sources package!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
#-------------------------------------------------------------------------------
          # Remove the epx_ordo.txt file if present, because it
          # contains only the names in the sources package and NOT those
          # of the retrieved includers.
          # A new, complete epx_ordo.txt will be automatically
          # generated when invoking epx_cmp without file names!
          # Same thing in case full recompilation is needed!
          #
          if (defined ($ifile) || defined ($full_compil)) {
            if ( -r "epx_ordo.txt" ) {
              print "\n===> Removing epx_ordo.txt file from sources package.\n";
              unlink "epx_ordo.txt";
            }
          }
#-------------------------------------------------------------------------------
          # Compile the sources
          #
          unlink "epx_cmp.err";
          system("epx_cmp -q -o");
          if ( -r "epx_cmp.err" ) {
            $errmsg = "Die epx_evol_start: compilation ERROR (epx_cmp)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#-------------------------------------------------------------------------------
        # Link and produce an executable module, if necessary. If there are
        # benchmarks but neither sources nor includes in evolution, then
        # no link is performed, but the standard executable is
        # copied locally under the name epx.exe.
        #
        if (defined($sfile) || defined($ifile) || defined($bfile)) {
          #
          # Benchs are necessary because this is not a "manuals-only"
          # evolution!
          #
          if (defined($sfile) || defined($ifile)) {
            print "\n===> Linking to produce a local library (libplex.lib)\n";
            print "\n===> and executable (epx.exe).\n";
            #
            unlink "epx_lk.err";
            if (defined ($full_compil)) {
              # Full recompilation has been done. Build up the local library
              # ex-novo, i.e. by ignoring the standard one (-L switch)
              system("epx_lk -L -o");
            }
            else {
              system("epx_lk -l -o");
            }
            if ( -r "epx_lk.err" ) {
              $errmsg = "Die epx_evol_start: ERROR in epx_lk!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          else {
            print "\n===> Copying standard executable to epx.exe\n";
#fc use copyux otherwise it prompts user: is it a file or directory name!
            system ("$copyux $stdexe epx.exe");
          }
#-------------------------------------------------------------------------------
          # Execute the benchmark test suite
          #
          print "\n===> Executing the benchmark test suite\n";
          #
          # Process the benchmarks package, if any
          #
          if (defined ($bfile)) {
            print "===> Gunzipping and untarring $bfile\n";
            system("gunzip $bfile");
            $bbase = $bfile; $bbase =~ s/\.tar\.gz$//;
            system("tar xvf $bbase.tar 2>>$errors");
            system("gzip -f -9 $bbase.tar");
            #
            # Verify that a proper history file is contained within the
            # benchmarks package
            #
            $bfileh = $bfile;
            $bfileh =~ s/\.tar\.gz$//;
            $bfileh =~ s/^B_/Cb/;
            $bfileh = "$bfileh\.hist";
            if (! -r $bfileh) {
              $errmsg =
              "History file $bfileh missing in benchmarks package!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          #
          if (! defined ($nobench)) {
            # Run the local tests (if any). Run also the standard
            # benchmark tests, if either the sources or the includes
            # (or both) have changed.
            #
            if (defined($sfile) || defined($ifile)) {
              $local = "";
            }
            else {
              $local = "-l";
            }
            #
            unlink "epx_test_benchmarks.err";
            system("epx_test_benchmarks $local");
            if ( -r "epx_test_benchmarks.err" ) {
              $errmsg = "Die epx_evol_start: ERROR in epx_test_benchmarks!\007\n
";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          else {
            print "\n===> NOT testing the benchmarks (-nobench switch)\n";
          }
        }
#-------------------------------------------------------------------------------
        # Update the includes library if necessary
        #
        if (defined($ifile)) {
          print "\n===> Updating the includes library\n";
          if (system ("$copy *.inc $incdir")) {
            $errmsg = "Die epx_evol_start: ERROR updating the includes!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#-------------------------------------------------------------------------------
        # Update the sources library if necessary
        #
        if (defined($sfile)) {
          print "\n===> Updating the sources library\n";
          if (system ("$copy *.ff $srcdir")) {
            $errmsg = "Die epx_evol_start: ERROR updating the sources!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#-------------------------------------------------------------------------------
        # Update the modules (.mod) library if necessary
        #
        if (defined($ifile) || defined($sfile)) {
          #
          # Verify that there are > 0 mod files, to avoid copy error
          #
          $nummod = `ls -l *.mod 2>>$errors | wc -l`;
          chop $nummod;
          $nummod =~ s/[ \t]//g;
          if ($nummod > 0) {
            print "\n===> Updating the modules library\n";
            if (system ("$copy *.mod $moddir")) {
              $errmsg = "Die epx_evol_start: ERROR updating the modules!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
        }
#-------------------------------------------------------------------------------
        # Update the objects (.lib) library if necessary
```

```
        #
        if (defined($ifile) || defined($sfile)) {
          print "\n===> Updating the objects library\n";
          if (system ("$copy libplex.lib $libdir")) {
            $errmsg = "Die epx_evol_start: ERROR updating the library!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#-------------------------------------------------------------------------------
        # Update the executable if necessary
        #
        if (defined($sfile) || defined($ifile)) {
          print "\n===> Updating the executable\n";
          if (system ("$copy epx.exe $stdexe")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the executable!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
#          #
#          # Update also the on-line executable
#          #
#          print "\n===> Update the on-line executable (see evol_putexe.log)\n";
##fc use copyux because of LOCAL copy!
#          system ("$copyux epx.exe europlexus.exe");
#          system ("gzip -f -9 europlexus.exe");
##          if (system ("$copy europlexus.exe.gz $wwwexe\\europlexus\.exe\.gz"))
{
##            $errmsg =
##            "Die epx_evol_start: ERROR updating the on-line executable!\007\n";
##            print "$errmsg";
##            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
##          }
#          system ("epx_ftp_putexe");
##
#          system ("$del europlexus.exe");
          #-----------------------------------------------------------------------
          # Generate also the QuickWin executable and update it
          if (! defined ($noqw)) {
            print "\n===> Linking to produce a local library (libplex.lib)\n";
            print "\n===> and QuickWin executable (epxqw.exe).\n";
            #
            unlink "epx_lk.err";
            if (defined ($full_compil)) {
              # Full recompilation has been done. Build up the local library
              # ex-novo, i.e. by ignoring the standard one (-L switch)
              system("epx_lk -L -o -w");
            }
            else {
              system("epx_lk -l -o -w");
            }
            if ( -r "epx_lk.err" ) {
              $errmsg = "Die epx_evol_start: ERROR in epx_lk -w!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            print "\n===> Updating the QuickWin executable\n";
            if (system ("$copy epxqw.exe $stdqwexe")) {
              $errmsg =
              "Die epx_evol_start: ERROR updating the QuickWin executable!\007\n
";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            # Update also the QuickWin on-line executable
            print "\n===> Update the QuickWin on-line executable\n";
            print " (see evol_putexe.log)\n";
#fc use copyux because of LOCAL copy!
            system ("$copyux epxqw.exe europlexusqw.exe");
            system ("gzip -f -9 europlexusqw.exe");
            system ("epx_ftp_putexe -w");
            system ("$del europlexusqw.exe");
          }
          else {
            print "\n===> NOT Updating the QuickWin version (-noqw switch)\n";
          }
          #-----------------------------------------------------------------------
          # Generate also the MPI executable and update it
          #
          if (! defined ($nompi)) {
            print "\n===> Updating the MPI executable\n";
            #
            # Remove work directory MPI if already existing
            if (-d "MPI") {
              print "\n  ===> Deleting the previous MPI directory\n";
# use rd instead of rmdir since in some cases rmdir seems to use
# GNU's rmdir instead of DOS rmdir (which is the one I want to use here)
#             system ("rmdir /s/q MPI >>$errors 2>>&1");
              system ("rd /s/q MPI >>$errors 2>>&1");
#
            }
            # Create a work directory MPI and go in it
            print "\n  ===> Creating the MPI directory\n";
            if (system ("mkdir MPI")) {
              $errmsg =
              "Die epx_evol_start: ERROR mkdir MPI!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            print "\n  ===> Changing to the MPI directory\n";
            if (system ("chdir MPI")) {
              $errmsg =
              "Die epx_evol_start: ERROR chdir MPI!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            if (system ("$copy ..\\*.ff .")) {
              $errmsg =
              "Die epx_evol_start: ERROR copying sources for MPI!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            if (defined($ifile)) {
              if (system ("$copy ..\\*.inc .")) {
                $errmsg =
                "Die epx_evol_start: ERROR copying includes for MPI!\007\n";
                print "$errmsg";
                &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
              }
            }
            print "\n===> Compiling sources for MPI version\n";
            system("epx_cmp -q -o -M");
            if ( -r "epx_cmp.err" ) {
              $errmsg = "Die epx_evol_start: MPI compilation ERROR (epx_cmp)!\00
7\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            print "\n===> Linking MPI version\n";
            if (defined ($full_compil)) {
              # Full recompilation has been done. Build up the local library
              # ex-novo, i.e. by ignoring the standard one (-L switch)
              system("epx_lk -L -o -M");
            }
            else {
              system("epx_lk -l -o -M");
            }
            if ( -r "epx_lk.err" ) {
              $errmsg = "Die epx_evol_start: MPI linkage ERROR in epx_lk!\007\n"
;
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            print "\n===> Updating the MPI executable\n";
            if (system ("$copy epx.exe $stdMPIexe")) {
              $errmsg =
              "Die epx_evol_start: ERROR updating the MPI executable!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            # Update also the MPI on-line executable
            print "\n===> Update the MPI on-line executable\n";
            print " (see evol_putexe.log)\n";
#fc use copyux because of LOCAL copy!
            system ("$copyux epx.exe europlexus_mpi.exe");
            system ("gzip -f -9 europlexus_mpi.exe");
            system ("epx_ftp_putexe -M");
            system ("$del europlexus_mpi.exe");
            # Update the modules (.mod) library if necessary
            #
            #
            # Verify that there are > 0 mod files, to avoid copy error
            #
            $nummod = `ls -l *.mod 2>>$errors | wc -l`;
            chop $nummod;
            $nummod =~ s/[ \t]//g;
            if ($nummod > 0) {
              print "\n===> Updating the MPI modules library\n";
              if (system ("$copy *.mod $MPImoddir")) {
                $errmsg = "Die epx_evol_start: ERROR updating the MPI modules!\0
07\n";
                print "$errmsg";
                &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
              }
            }
            # Update the objects (.lib) library if necessary
            #
            print "\n===> Updating the MPI objects library\n";
            if (system ("$copy libplex_MPI.lib $libdir")) {
              $errmsg = "Die epx_evol_start: ERROR updating the MPI library!\007
\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            # Return to parent directory
            print "\n  ===> Changing back to parent directory\n";
            if (system ("chdir ..")) {
              $errmsg =
              "Die epx_evol_start: ERROR chdir ..!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            # Clean up (remove entire temporary directory MPI)
            print "\n  ===> Removing the MPI work directory\n";
            system ("rd /s/q MPI >>$errors 2>>&1");
            #
          }
          else {
            print "\n===> NOT Updating the MPI executable (-nompi switch)\n";
          }
          #-----------------------------------------------------------------------
          # Generate also the -check executable and update it
          #
          if (! defined ($nocheck)) {
            print "\n===> Updating the -check executable\n";
            #
            # Remove work directory CHECK if already existing
            if (-d "CHECK") {
              print "\n  ===> Deleting the previous CHECK directory\n";
# use rd instead of rmdir since in some cases rmdir seems to use
# GNU's rmdir instead of DOS rmdir (which is the one I want to use here)
#             system ("rmdir /s/q CHECK >>$errors 2>>&1");
              system ("rd /s/q CHECK >>$errors 2>>&1");
#
            }
            # Create a work directory CHECK and go in it
            print "\n  ===> Creating the CHECK directory\n";
            if (system ("mkdir CHECK")) {
              $errmsg =
              "Die epx_evol_start: ERROR mkdir CHECK!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            print "\n  ===> Changing to the CHECK directory\n";
            if (system ("chdir CHECK")) {
              $errmsg =
              "Die epx_evol_start: ERROR chdir CHECK!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            if (system ("$copy ..\\*.ff .")) {
              $errmsg =
              "Die epx_evol_start: ERROR copying sources for CHECK!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
```

```
          }
          if (defined($ifile)) {
            if (system ("$copy ..\\*.inc .")) {
              $errmsg =
              "Die epx_evol_start: ERROR copying includes for CHECK!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          print "\n===> Compiling sources for CHECK version\n";
          system("epx_cmp -q -o -c");
          if ( -r "epx_cmp.err" ) {
            $errmsg = "Die epx_evol_start: CHECK compilation ERROR (epx_cmp)!\
007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "\n===> Linking CHECK version\n";
          if (defined ($full_compil)) {
            # Full recompilation has been done. Build up the local library
            # ex-novo, i.e. by ignoring the standard one (-L switch)
            system("epx_lk -L -o -c");
          }
          else {
            system("epx_lk -l -o -c");
          }
          if ( -r "epx_lk.err" ) {
            $errmsg = "Die epx_evol_start: CHECK linkage ERROR in epx_lk!\007\
n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "\n===> Updating the CHECK executable\n";
          if (system ("$copy epx.exe $stdCHECKexe")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the CHECK executable!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          # Update also the -check on-line executable
          print "\n===> Update the -check on-line executable\n";
          print " (see evol_putexe.log)\n";
#fc use copyux because of LOCAL copy!
          system ("$copyux epx.exe europlexus_check.exe");
          system ("gzip -f -9 europlexus_check.exe");
          system ("epx_ftp_putexe -c");
          system ("$del europlexus_check.exe");
          # Update the objects (.lib) library if necessary
          #
          print "\n===> Updating the CHECK objects library\n";
          if (system ("$copy libplex_check.lib $libdir")) {
            $errmsg = "Die epx_evol_start: ERROR updating the CHECK library!\0
07\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          # Return to parent directory
          print "\n  ===> Changing back to parent directory\n";
          if (system ("chdir ..")) {
            $errmsg =
            "Die epx_evol_start: ERROR chdir ..!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          # Clean up (remove entire temporary directory CHECK)
          print "\n  ===> Removing the CHECK work directory\n";
          system ("rd /s/q CHECK >>$errors 2>>&1");
          #
        }
        else {
          print "\n===> NOT Updating the -check executable (-nocheck switch)\n
";
        }
      }
#-------------------------------------------------------------------------------
      # Update the benchmarks library if necessary
      #
      if (defined($sfile) || defined($ifile) || defined($bfile)) {
        print "\n===> Updating the benchmarks\n";
        #
        # Input data file is mandatory
        #
        if (system ("$copy *.epx $bendir")) {
          $errmsg =
          "Die epx_evol_start: ERROR updating the benchmarks(1)!\007\n";
          print "$errmsg";
          &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
        }
        #
        # Mesh file is optional
        # Verify that there are > 0 msh files, to avoid copy error
        #
        $nummsh = `ls -l *.msh 2>>$errors | wc -l`;
        chop $nummsh;
        $nummsh =~ s/[ \t]//g;
        if ($nummsh > 0) {
          if (system ("$copy *.msh $bendir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the benchmarks(2)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
        #
        # Zip file (bm_*.zip) is optional
        # Verify that there are > 0 zip files, to avoid copy error
        #
        $numzip = `ls -l bm_*.zip 2>>$errors | wc -l`;
        chop $numzip;
        $numzip =~ s/[ \t]//g;
        if ($numzip > 0) {
          if (system ("$copy bm_*.zip $bendir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the benchmarks(3)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
        if (! defined ($nobench)) {
          #
          # Listing file is mandatory
          #
          if (system ("$copy *.listing $bendir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the benchmarks(4)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          #
          # PostScript file is optional
          # Verify that there are > 0 ps files, to avoid copy error
          #
          $numps = `ls -l *.ps 2>>$errors | wc -l`;
          chop $numps;
          $numps =~ s/[ \t]//g;
          if ($numps > 0) {
            if (system ("$copy *.ps $bendir")) {
              $errmsg =
              "Die epx_evol_start: ERROR updating the benchmarks(5)!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          #
          # BMP file(s) are optional
          # Verify that there are > 0 bmp files, to avoid copy error
          #
          $numbmp = `ls -l *.bmp 2>>$errors | wc -l`;
          chop $numbmp;
          $numbmp =~ s/[ \t]//g;
          if ($numbmp > 0) {
            if (system ("$copy *.bmp $bendir")) {
              $errmsg =
              "Die epx_evol_start: ERROR updating the benchmarks(6)!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          #
          # AVI file(s) are optional
          # Verify that there are > 0 avi files, to avoid copy error
          #
          $numavi = `ls -l *.avi 2>>$errors | wc -l`;
          chop $numavi;
          $numavi =~ s/[ \t]//g;
          if ($numavi > 0) {
            if (system ("$copy *.avi $bendir")) {
              $errmsg =
              "Die epx_evol_start: ERROR updating the benchmarks(7)!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          #
          # PVD file(s) are optional
          # Verify that there are > 0 pvd files, to avoid copy error
          #
          $numpvd = `ls -l *.pvd 2>>$errors | wc -l`;
          chop $numpvd;
          $numpvd =~ s/[ \t]//g;
          if ($numpvd > 0) {
            if (system ("$copy *.pvd $bendir")) {
              $errmsg =
              "Die epx_evol_start: ERROR updating the benchmarks(8)!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          #
          # VTU file(s) are optional
          # Verify that there are > 0 vtu files, to avoid copy error
          #
          $numvtu = `ls -l *.vtu 2>>$errors | wc -l`;
          chop $numvtu;
          $numvtu =~ s/[ \t]//g;
          if ($numvtu > 0) {
            if (system ("$copy *.vtu $bendir")) {
              $errmsg =
              "Die epx_evol_start: ERROR updating the benchmarks(9)!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
        }
        else {
          print "\n===> NOT testing the benchmarks (-nobench switch)\n";
        }
      }
#-------------------------------------------------------------------------------
      # Process the manuals package, if any
      #
      if (defined($mfile)) {
        print "\n===> Testing manuals\n";
        print "===> Gunzipping and untarring $mfile\n";
        system("gunzip $mfile");
        $mbase = $mfile; $mbase =~ s/\.tar\.gz$//;
        system("tar xvf $mbase.tar 2>>$errors");
        system("gzip -f -9 $mbase.tar");
        #
        # Verify that a proper history file is contained within the
        # manuals package
        #
        $mfileh = $mfile;
        $mfileh =~ s/\.tar\.gz$//;
        $mfileh =~ s/^M_/Cm/;
        $mfileh = "$mfileh\.hist";
        if (! -r $mfileh) {
          $errmsg = "History file $mfileh missing in manuals package!\007\n";
          print "$errmsg";
          &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
        }
#-------------------------------------------------------------------------------
        # Test the manuals
        #
        unlink "epx_test_manuals.err";
        system("epx_test_manuals");
```

```
          if ( -r "epx_test_manuals.err" ) {
            $errmsg = "Die epx_evol_start: ERROR in epx_test_manuals!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
#-------------------------------------------------------------------------------
          # Update the manuals library
          #
          print "\n===> Updating the manuals\n";
          if (system ("$copy *.ttx $mandir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the manuals(1)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  LaTeX sources updated.\n";
#
          if (system ("$copy *.tex $manfildir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the filtered manuals(1)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  LaTeX filtered sources updated.\n";
#
          if (system ("$copy manual.dvi $mandir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the manuals(2)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  DVI version updated.\n";
#
          if (system ("$copy manual.ps $mandir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the manuals(3)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  PostScript version updated.\n";
#
          if (system ("$copy manual.pdf $mandir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the manuals(4)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  PDF version updated.\n";
#
          if (system ("$copy manual.html $mandir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the manuals(5)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  HTML (Tth) version updated.\n";
#
          if (system ("$copy manual_h.html $mandir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the manuals(6)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  HTML (Hevea/Hacha) monolithic version updated.\n";
#
          if (system ("$copy hacha\\*.* $hacdir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the manuals(7)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  HTML (Hevea/Hacha) split version updated.\n";
#
          #
          # Update also the on-line manual (PDF + Hevea/Hacha split versions)
          #
#         if (system ("$copy manual.pdf $wwwman")) {
#           $errmsg =
#           "Die epx_evol_start: ERROR updating on-line PDF manual!\007\n";
#           print "$errmsg";
#           &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
#         }
#          system ("epx_ftp_putman");
#
#          print "  PDF on-line version update (see evol_putman.log).\n";
#
#         if (system ("$copy hacha\\*.* $wwwman")) {
#           $errmsg =
#           "Die epx_evol_start: ERROR updating on-line HTML manual!\007\n";
#           print "$errmsg";
#           &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
#         }
#          print "  HTML(Hacha) on-line version update (see evol_putman.log).\n"
;
        }
#-------------------------------------------------------------------------------
        # Process the validations package, if any
        #
        if (defined($vfile)) {
          print "\n===> Testing validations\n";
          print "===> Gunzipping and untarring $vfile\n";
          system("gunzip $vfile");
          $vbase = $vfile; $vbase =~ s/\.tar\.gz$//;
          system("tar xvf $vbase.tar 2>>$errors");
          system("gzip -f -9 $vbase.tar");
          #
          # Verify that a proper history file is contained within the
          # validations package
          #
          $vfileh = $vfile;
          $vfileh =~ s/\.tar\.gz$//;
          $vfileh =~ s/^V_/Cv/;
          $vfileh = "$vfileh\.hist";
          if (! -r $vfileh) {
            $errmsg = "History file $vfileh missing in validat. package!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
#-------------------------------------------------------------------------------
          # Update the validations library
          #
          print "\n===> Updating the validations\n";
          if (system ("$copy vl_*.vld $valdir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the validations(1)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  Validation description files (vl_*.vld) updated.\n";
#
          if (system ("$copy vl_*.zip $valdir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the validations(2)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  Validation package files (vl_*.zip) updated.\n";
        }
#-------------------------------------------------------------------------------
        # Process the history file(s), if any, and update the histories
        #
        if (defined($sfileh) || defined($bfileh) || defined($mfileh) ||
            defined($vfileh)) {
          print "\n===> Processing the history file(s)\n";
          unlink "epx_evol_histories.err";
          if (defined($sfileh)) {
            print "  Processing the sources/includes history file(s)\n";
            system("epx_evol_histories $sfileh");
            if ( -r "epx_evol_histories.err" ) {
              $errmsg =
              "Die epx_evol_start: ERROR in epx_evol_histories!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          if (defined($bfileh)) {
            print "  Processing the benchmarks history file(s)\n";
            system("epx_evol_histories $bfileh");
            if ( -r "epx_evol_histories.err" ) {
              $errmsg =
              "Die epx_evol_start: ERROR in epx_evol_histories!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          if (defined($mfileh)) {
            print "  Processing the manuals history file(s)\n";
            system("epx_evol_histories $mfileh");
            if ( -r "epx_evol_histories.err" ) {
              $errmsg =
              "Die epx_evol_start: ERROR in epx_evol_histories!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          if (defined($vfileh)) {
            print "  Processing the validations history file(s)\n";
            system("epx_evol_histories $vfileh");
            if ( -r "epx_evol_histories.err" ) {
              $errmsg =
              "Die epx_evol_start: ERROR in epx_evol_histories!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          print "Histories updating has successfully terminated.\n";
        }
#-------------------------------------------------------------------------------
        # Copy the resume file, the trace file, and the evolution package
        # files (if any) to the backup directory, then delete them from
        # the FTP directory.
        #
        system("$copy $dirr $bkpdir"); unlink $dirr;
        system("$copy $dirt $bkpdir"); unlink $dirt;
        if (defined($ifile)) {system("$copy $diri $bkpdir"); unlink $diri};
        if (defined($sfile)) {system("$copy $dirs $bkpdir"); unlink $dirs};
        if (defined($bfile)) {system("$copy $dirb $bkpdir"); unlink $dirb};
        if (defined($mfile)) {system("$copy $dirm $bkpdir"); unlink $dirm};
        if (defined($vfile)) {system("$copy $dirv $bkpdir"); unlink $dirv};
#-------------------------------------------------------------------------------
        # Reset the variables that are tested through "defined",
        # in case another evolution has to be performed
        #
        undef $ifile; undef $sfile; undef $bfile; undef $mfile; undef $vfile;
        undef $sfileh; undef $bfileh; undef $mfileh; undef $vfileh;
        undef $ifileunpacked;
#-------------------------------------------------------------------------------
        # The evolution is terminated:
        #   - increment the evolution index;
        #   - clean up the files in $evodir (.);
        #
        print "\n===> EVOLUTION N. $num SUCCESSFUL!\n";
        print "===> Incrementing the evolution index\n";
        #
        system ("echo $num >$evonumfile");
        $evonum = `cat $evonumfile`;
        chop $evonum;
        print "\n===> The current evolution index is  : \# $evonum\n";
        #
        $numok = $evonum + 1;
        print "===> The next evolution index must be: \# $numok\n";
        #
        print "\n===> Cleaning up the evolution directory\n";
        system("$del *.gz"); system("$del *.hist");          # Packages
        system("$del *.ff"); system("$del *.f");             # Sources
        system("$del *.inc");                                # Includes
        system("$del *.obj"); system("$del *.err");          # Compilation
        system("$del *.mod");
        system("$del *.exe"); system("$del *.lib");          # Link
        system("$del *.pdb");
        system("$del *.eco"); system("$del *.std");          # Run/benchs
        system("$del *.epx"); system("$del *.msh");
        system("$del *.listing"); system("$del *.ps");
        system("$del bm_*.log"); system("$del bm_*.zip");
        system("$del bm_*.plog"); system("$del bm_*.pin");
        system("$del bm_*.lks");
```

```
        system("$del bm_listing.lst");
        system("$del *.bmp"); system("$del *.ide");
        system("$del *.sau");
        system("$del vl_*.vld"); system("$del vl_*.zip");    # Validations
        system("$del bm_*.ali");                             # Outputs
        system("$del bm_*.alt");
        system("$del bm_*.inp");
        system("$del bm_*.k20");
        system("$del bm_*.k200"); system("$del bm_*.k2000");
        system("$del bm_*.mtv");
        system("$del *.pun");
        system("$del bm_*.puk");
        system("$del bm_*.mgr");
        system("$del bm_*.txt");
        system("$del bm_*.avi");
        system("$del bm_*.pvd");
        system("$del bm_*.vtu");
        system("$del bm_*.str");
        system("$del bm_*.savr");
        system("$del bm_*.tpl");
        system("$del bm_*.unv");
        system("$del bm_*.xpl");
        system("$del bm_*.poc");
        system("$del bm_stderr.lst");
        system("$del epx_ordo.txt");
        system("$del epx*.exp");
        system("$del *.u18");
        system("$del *.dat");
        system("$del *.inp"); system("$del *.mif");
        system("$del *.p10"); system("$del *.pov");
        system("$del fort.*"); system("$del *.k");
        system("$del *.ttx"); system("$del *.aux");          # Manuals
        system("$del *.tex");
        system("$del gbnotice.*"); system("$del manual.*");
        system("$del manual_h.*");
        system("$del epx_pass_*.log");
        system("$del tth.log");
        if (-d "hacha") {
#         system("$del hacha\\*.*");
#         system("$rmdir hacha");
#
# use rd instead of rmdir since in some cases rmdir seems to use
# GNU's rmdir instead of DOS rmdir (which is the one I want to use here)
#         system("rmdir /s/q hacha >>$errors 2>>&1");
          system("rd /s/q hacha >>$errors 2>>&1");
#
        }
        system("$del _*.*");                                 # Temporary
#
# Reset full compilation flag, in case of multiple evolutions !
        if (defined ($full_compil)) {
          undef $full_compil;
        }
#-------------------------------------------------------------------------------
        # LAST OPERATION: reset the evolution lock.
        #
        print "\n===> Removing the evolution lock\n";
        unlink "$evolck";
#-------------------------------------------------------------------------------
# Report user and system CPU times for this process and for its children
#
        ($userb, $systemb, $cuserb, $csystemb) = times;
        $user    = $userb    - $usera;
        $system  = $systemb  - $systema;
        $cuser   = $cuserb   - $cusera;
        $csystem = $csystemb - $csystema;
#
        print "\nThis process used the following CPU times:\n";
        print "User   : $user\n";
        print "System : $system\n";
        print "\nThe children of this process used the following CPU times:\n";
        print "Cuser  : $cuser\n";
        print "Csystem: $csystem\n\n";
#-------------------------------------------------------------------------------
# Get and print current date and time
#
        $time = time;
        $gmttime = time2str($time);
        $loctime = time2iso($time);
        print "===> JRC EUROPLEXUS evolution \# $lognum ended   on: $gmttime\n";
        print "===> Local date/time: $loctime\n";
#-------------------------------------------------------------------------------
        # Redirect STDOUT and STDERR to $logfile
        #
        close STDOUT;
        close STDERR;
        open STDOUT, ">>$logfile";
        open STDERR, ">>$logfile";
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
        $oldfh = select(STDOUT); $| = 1; select($oldfh);
        $oldfh = select(STDERR); $| = 1; select($oldfh);
#
        print "  Redirecting STDOUT and STDERR to $logfile.\n";
        print "\n===> EVOLUTION N. $num SUCCESSFUL at $loctime!\n";
#-------------------------------------------------------------------------------
        # Mail evolution log file to project partners
        #
#        print "\n===> Mailing log file $evolog to partners.\n\n";
#        unlink "epx_mail.err";
#        $stat = "OK";
#        system("epx_mail $stat $num $evolog");
#        if ( -r "epx_mail.err" ) {
#          die "Die epx_evol_start: ERROR in epx_mail!\007\n";
#        }
#-------------------------------------------------------------------------------
        # Move evolution log file to trace directory
        #
        print "\n===> Moving evolution log file $evolog\n";
        print "      to $trcdir.\n";
        system("$copy $evolog $trcdir"); unlink $evolog;
      }
      else {
        print "The evolution index is out of sequence: skip this evolution!\n";
      }
    }
  }
}
#-------------------------------------------------------------------------------
# Report user and system CPU times for this process and for its children
#
($user1, $system1, $cuser1, $csystem1) = times;
#
$user    = $user1    - $user0;
$system  = $system1  - $system0;
$cuser   = $cuser1   - $cuser0;
$csystem = $csystem1 - $csystem0;
#
print "\nThis process used the following CPU times:\n";
print "User   : $user\n";
print "System : $system\n";
print "\nThe children of this process used the following CPU times:\n";
print "Cuser  : $cuser\n";
print "Csystem: $csystem\n\n";
#-------------------------------------------------------------------------------
# Get and print current date and time
#
$time = time;                       # Machine time (seconds since epoch)
$gmttime = time2str($time);         # Universal (GMT) date and time
$loctime = time2iso($time);         # This requires function time2iso to be
                                    # exported in module HTTP\Date.pm (is not
                                    # exported by default!
print "===> EUROPLEXUS evolution(s) at JRC ended   on: $gmttime\n";
print "===> Local date/time: $loctime\n";
#-------------------------------------------------------------------------------
# Copy evol.log and evol_*.log files to trace directory
#
close STDOUT;
close STDERR;
system("$copy $logfile $trcdir");
system("$copy $ftplogfile $trcdir");
system("$copy $exelogfile $trcdir");
system("$copy $manlogfile $trcdir");
#
exit;


#-------------------------------------------------------------------------------
sub ERRMAIL {
  local ($logfile, $num, $evolog) = @_;
#
  close STDOUT;
  close STDERR;
  open STDOUT, ">>$logfile";
  open STDERR, ">>$logfile";
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
  $oldfh = select(STDOUT); $| = 1; select($oldfh);
  $oldfh = select(STDERR); $| = 1; select($oldfh);
#
  print "\n===> Mailing log file $evolog to partners.\n\n";
  unlink "epx_mail.err";
  $stat = "FAILED!!!";
  system("epx_mail $stat $num $evolog");
  if ( -r "epx_mail.err" ) {
    die "Die epx_evol_start: ERROR in epx_mail!\007\n";
  }
}
#-------------------------------------------------------------------------------
```

epx_evol_check.pl

```
#-------------------------------------------------------------------------------
# Run all benchmarks with -check version of the executable
#-------------------------------------------------------------------------------
# Modules needed
#
use HTTP::Date;
#-------------------------------------------------------------------------------
# Redirect STDOUT and STDERR to evol_check.log (Fixed path!!!!)
#
$logfile = "C:\\EUROPLEXUS\\Fromcentral\\evol_check.log";
open STDOUT, ">>$logfile";
open STDERR, ">>$logfile";
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Get and print current date and time
#
$time = time;                       # Machine time (seconds since epoch)
$gmttime = time2str($time);         # Universal (GMT) date and time
$loctime = time2iso($time);         # This requires function time2iso to be
                                    # exported in module HTTP\Date.pm (is not
                                    # exported by default!
#
print "\n\n";
print "=======================================";
print "=========================================\n";
print "===> EUROPLEXUS benchmarks suite with -check started on: $gmttime\n";
print "===> Local date/time: $loctime\n";
#-------------------------------------------------------------------------------
# Default values
#
$copy = "xcopy /Q /R /Y";
$del  = "rm -f";
#-------------------------------------------------------------------------------
$gnudir = $ENV{'GNUDIR'};
if ( ! defined $gnudir) {
  die "The GNUDIR environment variable is undefined!\007\n";
}
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 0
#
if ($#ARGV > -1) {                  # Index of last argument must be = -1
  die "Usage: epx_evol_check\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
```

```
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  die "The EUROPLEXUS environment variable is undefined!\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  die "The EUROPLEXUS directory: $epx does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Set and check existence of auxiliary directories
#
$chkdir = "$epx\\Check";             # Directory used for running the benchs
if ( ! -d $chkdir) {
  die "The EUROPLEXUS directory: $chkdir does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Make sure we are in $chkdir (so we may use . instead of $chkdir)
#
$curdir = `pwd`;
chop $curdir;
if ( $curdir ne $chkdir ) {
  print "\nChanging directory to $chkdir.\n";
  chdir "$chkdir" ||
  die "Can't chdir to $chkdir!\007\n";
}
#-------------------------------------------------------------------------------
# Clean up previous benchmarks if any
#
print "\n===> Cleaning up the evolution directory\n";
system("$del *.*");
#-------------------------------------------------------------------------------
# Copy -check executable to current directory
#
print "\n===> Copying -check executable locally\n";
system("$copy $epx\\Exe\\Europlexus_check.exe .");
#-------------------------------------------------------------------------------
# Execute all benchmarks
#
print "\n===> Executing all benchmarks\n";
system("epx_test_benchmarks -e Europlexus_check.exe >_errors.log 2>&1");
#system("epx_bench -e europlexus_check.exe bm_flu_momt08 >_errors.log 2>&1");
#system("epx_get -b bm_flu_momt08");
#system("epx_test_benchmarks -e Europlexus_check.exe -l >_errors.log 2>&1");
#-------------------------------------------------------------------------------
# Get and print current date and time
#
$time = time;                        # Machine time (seconds since epoch)
$gmttime = time2str($time);          # Universal (GMT) date and time
$loctime = time2iso($time);          # This requires function time2iso to be
                                     # exported in module HTTP\Date.pm (is not
                                     # exported by default!
print "\n===> EUROPLEXUS benchmarks suite with -check ended on: $gmttime\n";
print "===> Local date/time: $loctime\n";
print "=======================================";
print "=======================================\n";
#-------------------------------------------------------------------------------
close STDOUT;
close STDERR;
#
exit;
```

epx_evol_files.pl

```
#
# Evolve EUROPLEXUS include and source file(s)
#-------------------------------------------------------------------------------
# Default values
#
$copy = "copy";                          # Alternative: use cp -p
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 0
#
if ($#ARGV != -1) {                      # Index of last argument must be = -1
  die "Usage: epx_evol_files\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  die "The EUROPLEXUS environment variable is undefined!\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  die "The EUROPLEXUS directory: $epx does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Evolve object files (library), if any
#
$lib = "$epx\library\Libplex.lib";
if ( ! -r $lib) {
  die "The EUROPLEXUS object library: $lib does not exist!\007\n";
}
system ("LIB $lib *.obj");
#-------------------------------------------------------------------------------
# Evolve include files, if any
#
$inc = "$epx\include";
if ( ! -d $inc) {
  die "The EUROPLEXUS include directory: $inc does not exist!\007\n";
}
system ("$copy *.inc $inc");
#-------------------------------------------------------------------------------
# Evolve module (.mod) files, if any
#
$mod = "$epx\module";
if ( ! -d $mod) {
  die "The EUROPLEXUS module directory: $mod does not exist!\007\n";
}
system ("$copy *.mod $mod");
#-------------------------------------------------------------------------------
# Evolve source files, if any
#
$ff = "$epx\source";
if ( ! -d $ff) {
  die "The EUROPLEXUS source directory: $ff does not exist!\007\n";
}
system ("$copy *.ff $ff");
#-------------------------------------------------------------------------------
#
exit;
```

epx_evol_hist.pl

```
#
# Evolve EUROPLEXUS history file(s)
#-------------------------------------------------------------------------------
# Default values
#
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 1
#
if ($#ARGV != 0) {                       # Index of last argument must be = 0
  die "Usage: epx_evol_hist file\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  die "The EUROPLEXUS environment variable is undefined!\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  die "The EUROPLEXUS directory: $epx does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS history directory exists
#
$epxh = "$epx\\history";
if ( ! -d $epxh ) {
  die "The EUROPLEXUS history directory: $epxh does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# The argument is the global history file name
#
$his = $ARGV[0];
#-------------------------------------------------------------------------------
# Remove file name extension if needed
#
$base = $his;
$base =~ s/\.hist//;
$his = "$base.hist";
#-------------------------------------------------------------------------------
# Check that the global history file exists
#
if ( ! -r $his ) {
  die "The global history file: $his does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Fold the global history file on a copy
#
system("fold -s $his >_TEMP_.hist");
#-------------------------------------------------------------------------------
# Open the global history file and (logically) split it into pieces which
# are appended to the single history files
#
open (HIST, _TEMP_.hist);
#print "File $his opened\n";
while (<HIST>) {
  s/\r$//; # The stupid "fold" (sometimes?) adds an extra \r at the end of
           # each line (from Aug. 2005 on), so let's remove it if present,
           # otherwise the following pattern recognitions don't work!
  #print "LINE: $_";
  if ( m/^=== Evolution des sources d\'EUROPLEXUS du/ ) {
    #-----------------------------------------------------------------------
    # Line matches "first line" pattern (text to be used for includes)
    #
    if ( ! defined ($inchis)) {
      $inchis = $_;
    }
    else {
      die "First-line pattern matched more than once!\007\n";
    }
  }
  elsif ( m/^[ \t]*\w+\.INC[ \t]*$/ ) {
    #-----------------------------------------------------------------------
    # Line contains name of include file that has been evoluted.
    # Append $inchis to include history file
    #
    if ( ! defined ($inchis)) {
      die "Dont know what to write to include history file!\007\n";
    }
    else {
      chop;
      $fbase = $_;
      $fbase =~ s/ //g;
      $fbase =~ s/\t//g;
      $fbase =~ s/\.INC//;
      $file = "$epxh\\$fbase.his";
      print "Append to include history file: $file\n";
      open (FILE, ">> $file");
      print(FILE $inchis);
      close FILE;
    }
  }
  elsif ( m/^[\*C] .*\x2F..\x2F.....:..:.. *$/ ) {
    #-----------------------------------------------------------------------
    # Line contains "old" header (without evolution number at the end).
    # Extract file name (2nd line field) and append line
    # to the corresponding history file
    #
    #print "Header line read:\n";
    #@fields = split(/[ \t]+/, $_);
    #$name = @fields[1];
    ($comm,$name,$status,$owner,$date,$time) = split(/[ \t]+/, $_);
```

```
    #print "Comm   = $comm\n";
    #print "Name   = $name\n";
    #print "Status = $status\n";
    #print "Owner  = $owner\n";
    #print "Date   = $date\n";
    #print "Time   = $time\n";
    $name =~ tr/A-Z/a-z/;
    $file = "$epxh\\$name.his";
    print "Append to source? history file: $file\n";
    if (defined ($opened)) {
      close FILE;
      undef $opened;
    }
    open (FILE, ">> $file");
    $opened = 1;
    print(FILE $_);
  }
  elsif ( m/^[\*C] .*\x2F..\x2F.....:..:.. #\d\d\d\d *$/ ) {
    #-------------------------------------------------------------------------------
    # Line contains "new" header (with evolution number at the end).
    # Extract file name (2nd line field) and append line
    # to the corresponding history file
    #
    #print "Header line read:\n";
    #@fields = split(/[ \t]+/, $_);
    #$name = @fields[1];
    ($comm,$name,$status,$owner,$date,$time,$evo) = split(/[ \t]+/, $_);
    #print "Comm   = $comm\n";
    #print "Name   = $name\n";
    #print "Status = $status\n";
    #print "Owner  = $owner\n";
    #print "Date   = $date\n";
    #print "Time   = $time\n";
    #print "Evo#   = $evo\n";
    $name =~ tr/A-Z/a-z/;
    $file = "$epxh\\$name.his";
    print "Append to source? history file: $file\n";
    if (defined ($opened)) {
      close FILE;
      undef $opened;
    }
    open (FILE, ">> $file");
    $opened = 1;
    print(FILE $_);
  }
  elsif ( m/^[ \t]*\n/ ) {
    #print "Blank line read.\n";
    if (defined ($opened)) {
      close FILE;
      undef $opened;
    }
  }
  else {
    #print "Comment line read.\n";
    if (defined ($opened)) {
      print(FILE $_);
    }
    else {
      print "WARNING: dont know where to write the comment line:\n$_\n\007";
    }
  }
}
close HIST;
system("del _TEMP_.hist");
#
exit;
```

epx_evol_histories.pl

```
#-------------------------------------------------------------------------------
# Evolve EUROPLEXUS history file(s)
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$errfil = "epx_evol_histories.err";
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 1
#
if ($#ARGV != 0) {                      # Index of last argument must be = 0
  $errmsg = "Usage: epx_evol_histories file\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  $errmsg = "The EUROPLEXUS environment variable is undefined!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  $errmsg = "The EUROPLEXUS directory: $epx does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS history directory exists
#
$epxh = "$epx\\history";
if ( ! -d $epxh ) {
  $errmsg = "The EUROPLEXUS history directory: $epxh does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# The argument is the global history file name
#
$his = $ARGV[0];
#-------------------------------------------------------------------------------
# Remove file name extension if needed
#
$base = $his;
$base =~ s/\.hist//;
$his = "$base.hist";
#-------------------------------------------------------------------------------
# Check that the global history file exists
#
if ( ! -r $his ) {
  $errmsg = "The global history file: $his does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Open the global history file by folding it at spaces before 80 columns
# and (logically) split it into pieces which
# are appended to the single history files
#
open (HIST, "fold -s $his |");
#print "File $his opened\n";
while (<HIST>) {
  s/\r$//; # The stupid "fold" (sometimes?) adds an extra \r at the end of
           # each line (from Aug. 2005 on), so let's remove it if present,
           # otherwise the following pattern recognitions don't work!
  #print "LINE: $_";
  if ( m/^== Evolution EUROPLEXUS \x23\d\d\d\d du \d\d\x2F\d\d\x2F\d\d a / ) {
    #-----------------------------------------------------------------------
    # Line matches "first line" pattern
    #
    if ( ! defined ($inchis)) {
      ($c1,$c2,$c3,$cindex,$c5,$date,$c7,$time,$c9,$type,$cpindex,$c12) =
        split(/[ \t]+/, $_);
      $inchis = $_;
      #
      $index0 = $cindex; $index0 =~ s/^\#//;          # Evolution index NNNN
      $index = $index0; $index =~ s/^0+//;            # Same without leading 0s
      #
      $type =~ tr/a-z/A-Z/;
      #
      $pindex0 = $cpindex; $pindex0 =~ s/^\#//;       # Package index NNNN
      $pindex = $pindex0; $pindex =~ s/^0+//;         # Same without leading 0s
      print "  $_";
      print "  Evolution index: $index.\n";
      print "  Evolution date : $date.\n";
      print "  Evolution time : $time.\n";
      print "  Package   type : $type.\n";
      print "  Package   index: $pindex.\n";
    }
    else {
      $errmsg = "First-line pattern matched more than once!\007\n";
      &ERRFIL ($errfil, $errmsg); die "$errmsg";
    }
  }
  elsif ( m/^[ \t]*\w+\.INC[ \t]*$/ ) {
    #-----------------------------------------------------------------------
    # Line contains name of include file that has been evoluted.
    # Append $inchis to include history file
    #
    if ( ! defined ($inchis)) {
      $errmsg = "Dont know what to write to include history file!\007\n";
      &ERRFIL ($errfil, $errmsg); die "$errmsg";
    }
    else {
      chop;
      $fbase = $_;
      $fbase =~ s/ //g;
      $fbase =~ s/\t//g;
      $fbase =~ s/\.INC//;
      $file = "$epxh\\$fbase.his";
      print "Append to include history file: $file.\n";
      open (FILE, ">> $file");
      print(FILE $inchis);
      close FILE;
    }
  }
  elsif ( m/^[\*\$\%C\ß] .*\x2F..\x2F.....:..:.. *$/ ) {
    #-----------------------------------------------------------------------
    # Line contains "old" header (without evo # at the end):
    #   - Verify that the header type (source, bench, manual or validation)
    #     corresponds to $type
    #   - Extract file name (2nd line field) and append line
    #     to the corresponding history file
    #
    #print "Header line read:\n";
    #@fields = split(/[ \t]+/, $_);
    #$name = @fields[1];
    ($comm,$name,$status,$owner,$date,$time) = split(/[ \t]+/, $_);
    #print "Comm   = $comm\n";
    #print "Name   = $name\n";
    #print "Status = $status\n";
    #print "Owner  = $owner\n";
    #print "Date   = $date\n";
    #print "Time   = $time\n";
    $name =~ tr/A-Z/a-z/;
    $file = "$epxh\\$name.his";
    $rtype = "SOURCES";
    if ( m/^\$ BM/ ) {
      $rtype = "BENCHS";
    }
    elsif ( m/^\ß VL/ ) {
      $rtype = "VALIDATE";
    }
    elsif ( m/^\%/ ) {
      $rtype = "MANUAL";
    }
    if (! ($type eq $rtype)) {
      $errmsg = "Declared type is $type, read type is $rtype!\007\nLine: $_\n";
      &ERRFIL ($errfil, $errmsg); die "$errmsg";
    }
    print "Append to $rtype history file: $file.\n";
    if (defined ($opened)) {
      close FILE;
      undef $opened;
    }
    open (FILE, ">> $file");
    $opened = 1;
    print(FILE $_);
  }
  elsif ( m/^[\*\$\%C\ß] .*\x2F..\x2F.....:..:.. #\d\d\d\d *$/ ) {
    #-----------------------------------------------------------------------
```

```
    # Line contains "new" header (with evo # at the end):
    #   - Verify that the header type (source, bench, manual or validation)
    #     corresponds to $type
    #   - Extract file name (2nd line field) and append line
    #     to the corresponding history file
    #
    #print "Header line read:\n";
    #@fields = split(/[ \t]+/, $_);
    #$name = @fields[1];
    ($comm,$name,$status,$owner,$date,$time,$evo) = split(/[ \t]+/, $_);
    #print "Comm   = $comm\n";
    #print "Name   = $name\n";
    #print "Status = $status\n";
    #print "Owner  = $owner\n";
    #print "Date   = $date\n";
    #print "Time   = $time\n";
    #print "Evo#   = $evo\n";
    $name =~ tr/A-Z/a-z/;
    $file = "$epxh\\$name.his";
    $rtype = "SOURCES";
    if ( m/^\$ BM/ ) {
      $rtype = "BENCHS";
    }
    elsif ( m/^\$ VL/ ) {
      $rtype = "VALIDATE";
    }
    elsif ( m/^\$/ ) {
      $rtype = "MANUAL";
    }
    if (! ($type eq $rtype)) {
      $errmsg = "Declared type is $type, read type is $rtype!\007\nLine: $_\n";
      &ERRFIL ($errfil, $errmsg); die "$errmsg";
    }
    print "Append to $rtype history file: $file.\n";
    if (defined ($opened)) {
      close FILE;
      undef $opened;
    }
    open (FILE, ">> $file");
    $opened = 1;
    print(FILE $_);
  }
  elsif ( m/^[ \t]*\n/ ) {
    #print "Blank line read.\n";
    if (defined ($opened)) {
      close FILE;
      undef $opened;
    }
  }
  else {
    #print "Comment line read.\n";
    if (defined ($opened)) {
      print(FILE $_);
    }
    else {
      if ( ! m/^==================================+$/ ) {
        print "WARNING: dont know where to write the comment line:\n$_\n";
      }
    }
  }
}
close HIST;
#
exit;


#------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#------------------------------------------------------------------------------
```

epx_evol_start.pl

```
#------------------------------------------------------------------------------
# Start evolution of EUROPLEXUS files. All evolution packages present in the
# FTP directory are treated. If more than one evolution package set is
# present, they are applied in order.
#------------------------------------------------------------------------------
# Modules needed
#
use HTTP::Date;
#------------------------------------------------------------------------------
# Redirect STDOUT and STDERR to evol.log (Fixed path!!!!)
#
$logfile = "C:\\EUROPLEXUS\\Fromcentral\\evol.log";
open STDOUT, ">>$logfile";
open STDERR, ">>$logfile";
#------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#------------------------------------------------------------------------------
# Get and print current date and time
#
$time = time;                       # Machine time (seconds since epoch)
$gmttime = time2str($time);         # Universal (GMT) date and time
$loctime = time2iso($time);         # This requires function time2iso to be
                                    # exported in module HTTP\Date.pm (is not
                                    # exported by default!
print "\n\n";
print "=======================================";
print "=======================================\n";
print "===> EUROPLEXUS evolution(s) at JRC started on: $gmttime\n";
print "===> Local date/time: $loctime\n";
#------------------------------------------------------------------------------
# Initialize times
#
($user0, $system0, $cuser0, $csystem0) = times;
#------------------------------------------------------------------------------
# Default values
#
#$ftpdir = "C:\\IIS\\FTP\\EUROPLEXUS"; # Directory receiving FTP files
$ftpdir = "C:\\EUROPLEXUS\\FTP"; # Directory receiving FTP files
#
$wwwdir = "C:\\WWW\\EUROPLEXUS"; # Directory where to put on-line files
$wwwman = "$wwwdir\\MANUAL";     # Directory where to put on-line manual
$wwwexe = "$wwwdir\\EXE";        # Directory where to put on-line executable
#
#fc 02 May 2005
$copyux = "cp -p -f";
$copy = "xcopy /Q /R /Y /I";
#fc 02 May 2005
$del  = "rm -f";
#------------------------------------------------------------------------------
$gnudir = $ENV{'GNUDIR'};
if ( ! defined $gnudir) {
  die "The GNUDIR environment variable is undefined!\007\n";
}
$rmdir = "$gnudir\\rmdir.exe";   # Full path needed! Else uses MS-DOS's rmdir!
$errors = "C:\\EUROPLEXUS\\Fromcentral\\errors.txt";
#------------------------------------------------------------------------------
# Check number of arguments, must be = 0 or 1 or 2 or 3 or 4 or 5
#
if ($#ARGV > 3) { # Index of last argument must be = -1 or 0 or 1 or 2 or 3 or 4
  die "Usage: epx_evol_start [-p] [-noqw] [-nompi] [-nocheck] [-fullcmp]\007\n";
}
#------------------------------------------------------------------------------
# Process optional switches:
#       -p       patch (do not get evolution files by ftp)
#       -noqw    do not update the QuickWin executable
#       -nompi   do not update the MPI executable
#       -nocheck do not update the -check executable
#       -fullcmp full compilation (recompile all sources)
#
#print "Index of last arg is: $#ARGV.\n";
$nar = $#ARGV + 1;
#print "No. of args is: $nar.\n";
if ($#ARGV >= 0) { # There is at least one argument
  while ($ARGV[0] =~ /^-/) {
    $_ = shift;
    $nar = $nar - 1;
    if (/^-p(.*)/) {
      $patch = "yes";
      print "Command line switch: patch\n";
    }
    elsif (/^-noqw(.*)/) {
      $noqw = "yes";
      print "Command line switch: noqw\n";
    }
    elsif (/^-nompi(.*)/) {
      $nompi = "yes";
      print "Command line switch: nompi\n";
    }
    elsif (/^-nocheck(.*)/) {
      $nocheck = "yes";
      print "Command line switch: nocheck\n";
    }
    elsif (/^-fullcmp(.*)/) {
      $full_compil = "yes";
      print "Command line switch: fullcmp\n";
    }
    else {
      die "ERROR: unknown switch: $_\n";
    }
  }
}
#print "Index of last arg is: $#ARGV.\n";
#print "No. of args is: $nar.\n";
#------------------------------------------------------------------------------
# Get any evolution files by ftp and copy them to the local directory ($ftpdir)
#
$ftplogfile = "C:\\EUROPLEXUS\\Fromcentral\\evol_getfiles.log";
$exelogfile = "C:\\EUROPLEXUS\\Fromcentral\\evol_putexe.log";
$manlogfile = "C:\\EUROPLEXUS\\Fromcentral\\evol_putman.log";
#
if (! defined $patch) {
  print "\n===> Getting any evolution files from central mirror site.\n";
  print "     (see evol_getfiles.log)\n";
  system ("epx_ftp_getfiles");
}
else {
  print "\n===> Not getting any evol. files (-p option).\n";
}
#------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  die "The EUROPLEXUS environment variable is undefined!\007\n";
}
#------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  die "The EUROPLEXUS directory: $epx does not exist!\007\n";
}
#------------------------------------------------------------------------------
# Check existence of evolution version file, and NON-existence of
# evolution lock file
#
$evolck = "$epx\\evolution.lck";           # Evolution lock file (blocks
                                           # successive evols in case of errs)
$evonumfile = "$epx\\VERSION.txt";         # Evolution version file
if (! -r $evonumfile) {
  die "The evolution number file $evonumfile does not exist!\007\n";
}
if (-r $evolck) {
  die "Evolution is stopped because the lock file $evolck exists!\007\n";
}
#------------------------------------------------------------------------------
# Set and check existence of auxiliary directories
#
$bkpdir = "$epx\\Backup";                  # Directory holding backup info
$evodir = "$epx\\Fromcentral";             # Directory used for evolution
if ( ! -d $bkpdir) {
  die "The EUROPLEXUS directory: $bkpdir does not exist!\007\n";
}
if ( ! -d $evodir) {
  die "The EUROPLEXUS directory: $evodir does not exist!\007\n";
```

```
}
#-------------------------------------------------------------------------------
# Make sure we are in $evodir (so we may use . instead of $evodir)
#
$curdir = `pwd`;
chop $curdir;
if ( $curdir ne $evodir ) {
  print "\nChanging directory to $evodir.\n";
  chdir "$evodir" ||
  die "Can't chdir to $evodir!\007\n";
}
#-------------------------------------------------------------------------------
# Set and check existence of target directories
#
$srcdir = "$epx\\source";
$incdir = "$epx\\include";
$libdir = "$epx\\library";
$mandir = "$epx\\manual";
$manfildir = "$epx\\manual_filtered";
$hacdir = "$epx\\manual\\hacha";
$bendir = "$epx\\bench";
$hisdir = "$epx\\history";
$exedir = "$epx\\exe";
$trcdir = "$epx\\trace";
$moddir = "$epx\\module";
$MPImoddir = "$epx\\module_mpi";
$valdir = "$epx\\validate";
#
if ( ! -d $srcdir) {
  die "The EUROPLEXUS directory: $srcdir does not exist!\007\n";
}
if ( ! -d $incdir) {
  die "The EUROPLEXUS directory: $incdir does not exist!\007\n";
}
if ( ! -d $libdir) {
  die "The EUROPLEXUS directory: $libdir does not exist!\007\n";
}
if ( ! -d $mandir) {
  die "The EUROPLEXUS directory: $mandir does not exist!\007\n";
}
if ( ! -d $manfildir) {
  die "The EUROPLEXUS directory: $manfildir does not exist!\007\n";
}
if ( ! -d $hacdir) {
  die "The EUROPLEXUS directory: $hacdir does not exist!\007\n";
}
if ( ! -d $bendir) {
  die "The EUROPLEXUS directory: $bendir does not exist!\007\n";
}
if ( ! -d $hisdir) {
  die "The EUROPLEXUS directory: $hisdir does not exist!\007\n";
}
if ( ! -d $exedir) {
  die "The EUROPLEXUS directory: $exedir does not exist!\007\n";
}
if ( ! -d $trcdir) {
  die "The EUROPLEXUS directory: $trcdir does not exist!\007\n";
}
if ( ! -d $moddir) {
  die "The EUROPLEXUS directory: $moddir does not exist!\007\n";
}
if ( ! -d $valdir) {
  die "The EUROPLEXUS directory: $valdir does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Set and check existence of objects library
#
$objlib = "$libdir\\libplex\.lib";
if ( ! -r $objlib) {
  die "The EUROPLEXUS objects library: $objlib does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Set and check existence of the standard executable
#
$stdexe = "$exedir\\europlexus\.exe";
if ( ! -r $stdexe) {
  die "The EUROPLEXUS standard executable: $stdexe does not exist!\007\n";
}
# Set also the QuickWin executable
$stdqwexe = "$exedir\\europlexusqw\.exe";
# Set also the MPI executable
$stdMPIexe = "$exedir\\europlexus_mpi\.exe";
# Set also the -check executable
$stdCHECKexe = "$exedir\\europlexus_check\.exe";
#-------------------------------------------------------------------------------
# Read the number of the previous evolution
#
$evonum = `cat $evonumfile`;
chop $evonum;
print "\n===> The current evolution index is  : \# $evonum\n";
$numok = $evonum + 1;
print "===> The next evolution index must be: \# $numok\n";
#-------------------------------------------------------------------------------
# Treat all the evolution "resume" files, i.e., files of the
# form R_nnnnxjjmmmyy.txt in $ftpdir. These contain the list of
# evolution packages relative to each evolution, i.e names of files of the
# form [ISBMV]_NNNNxDDMMMYY.tar.gz that must also be in $ftpdir.
#
opendir(DIR, $ftpdir) || die "Can't open $ftpdir\n";
@FileNames = sort readdir(DIR);
closedir(DIR);
#
foreach $file (@FileNames) {
  $_ = $file;
  $dirf = "$ftpdir\\$file";
  if ( -r $dirf ) {
#-------------------------------------------------------------------------------
    # NOTE: the preceding test on file existence is necessary, because
    # treatment of a file in the list involves treatment (and deletion)
    # of all its matching files, if any!
    #
    if ( m/^R_\d\d\d\dx\d\d[A-Z][a-z][a-z]\d\d.txt$/ ) {
      print "\n===> Evolution resume file found: $file\n";
#-------------------------------------------------------------------------------
      # Evolution resume file found: open it and read names of
      # the associated evolution packages (I_, S_, B_, M_, V_), of which
      # zero to five must be listed, and which must be present in the
      # FTP directory.
      #
      # The case zero corresponds to an evolution which is empty because
      # it failed at the central mirror site. In this case the trace is
      # sent anyway (it will contain the error message). The only effects
      # on the mirror site are in this case:
      #   - the evolution counter is incremented;
      #   - the resume (R_) and trace (T_) files are moved to $trcdir
      #
      $rfile = $file;
      $rbase = $rfile; $rbase =~ s/\.txt$//;
      #
      $numdate = $rbase; $numdate =~ s/^R_//;     # Number & date (NNNNxDDMMMYY)
      $date = $numdate; $date =~ s/^.....//;      # Date (DDMMMYY)
      $num = $numdate; $num =~ s/x.......$//;     # Number (with leading 0s)
      $lognum = $num;
      $num =~ s/^0*//;                            # Number (no leading 0s)
      print "The evolution index is: $num\n";
      #
#-------------------------------------------------------------------------------
      # Is the index of this evolution the right one?
      #
      if ($num == $numok) {
        #
        $evolog = "$evodir\\$lognum\.log";
        #
        print "\n===> The evolution index is OK: the evolution starts.\n";
        print "\n  Redirecting STDOUT and STDERR to $evolog\n     ...\n";
#-------------------------------------------------------------------------------
        # Redirect STDOUT and STDERR to this evolution's log file ($evolog)
        #
        close STDOUT;
        close STDERR;
        open STDOUT, ">>$evolog";
        open STDERR, ">>$evolog";
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
        $oldfh = select(STDOUT); $| = 1; select($oldfh);
        $oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Get and print current date and time
#
        $time = time;
        $gmttime = time2str($time);
        $loctime = time2iso($time);
        # add 8 initial dummy lines
        # to avoid that possible error messages
        # may overwrite first part of log file
        print "============================================================\n";
        print "============================================================\n";
        print "============================================================\n";
        print "============================================================\n";
        print "============================================================\n";
        print "============================================================\n";
        print "============================================================\n";
        print "============================================================\n";
        print "===> JRC EUROPLEXUS evolution \# $lognum started on: $gmttime\n";
        print "===> Local date/time: $loctime\n";
#-------------------------------------------------------------------------------
# Initialize times
#
        ($usera, $systema, $cusera, $csystema) = times;
#-------------------------------------------------------------------------------
        # Set the lock file: it will be reset only when the evolution
        # is successfully terminated
        #
        system("echo $num >$evolck");
        print "\n===> The lock file has been set\n\n";
        #
        print "This evolution should contain the following packages:\n";
        print "  Trace file (mandatory)\n";
        #
        $dirr = "$ftpdir\\$rfile";
        #
        open (RESUME, $dirr);
        while (<RESUME>) {
          chop;
          if ( m/^[ISBMV]_\d\d\d\dx\d\d[A-Z][a-z][a-z]\d\d.tar.gz$/ ) {
            $entry = $_;
            $entry =~ s/^._.....//;
            $entry =~ s/\.tar\.gz$//;
            if ($entry eq $date) {
              #
              # The entry in the resume file matches the evolution date
              #
              if    ( m/^I/ ) {$ifile = $_; print "  Includes package.\n"}
              elsif ( m/^S/ ) {$sfile = $_; print "  Sources package.\n"}
              elsif ( m/^B/ ) {$bfile = $_; print "  Benchmarks package.\n"}
              elsif ( m/^M/ ) {$mfile = $_; print "  Manuals package.\n"}
              elsif ( m/^V/ ) {$vfile = $_; print "  Validations package.\n"}
              else {
                $errmsg =
                "The resume file contains an ambiguous name: $_!\007\n";
                print "$errmsg";
                &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
              }
            }
          }
        }
        close RESUME;
#-------------------------------------------------------------------------------
        # Verify that the trace file corresponding to the resume file
        # is present in $ftpdir
        #
        print "It contains:\n";
        $tfile = $rfile; $tfile =~ s/^R/T/;
        $dirt = "$ftpdir\\$tfile";
        #
        if (! -r $dirt) {
          $errmsg = "The trace file $tfile missing in $ftpdir!\007\n";
          print "$errmsg";
          &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
        }
        else {
          print "  Trace file $dirt present.\n";
        }
#-------------------------------------------------------------------------------
        # Copy the existing packages (I_, S_, B_, M_, V_), if any,
```

```
        # to the evolution directory.
        #
        if (defined($ifile)) {
          $diri = "$ftpdir\\$ifile";
          if ( -r $diri ) {
            print "  Includes package $diri present.\n";
            system("$copy $diri $evodir");
          }
          else {
            $errmsg = "File $ifile listed in $rfile missing in $ftpdir!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#-------------------------------------------------------------------------------
        if (defined($sfile)) {
          $dirs = "$ftpdir\\$sfile";
          if ( -r $dirs ) {
            print "  Sources package $dirs present.\n";
            system("$copy $dirs $evodir");
          }
          else {
            $errmsg = "File $sfile listed in $rfile missing in $ftpdir!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#-------------------------------------------------------------------------------
        if (defined($mfile)) {
          $dirm = "$ftpdir\\$mfile";
          if ( -r $dirm ) {
            print "  Manuals package $dirm present.\n";
            system("$copy $dirm $evodir");
          }
          else {
            $errmsg = "File $mfile listed in $rfile missing in $ftpdir!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#-------------------------------------------------------------------------------
        if (defined($bfile)) {
          $dirb = "$ftpdir\\$bfile";
          if ( -r $dirb ) {
            print "  Benchmarks package $dirb present\n";
            system("$copy $dirb $evodir");
          }
          else {
            $errmsg = "File $bfile listed in $rfile missing in $ftpdir!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#-------------------------------------------------------------------------------
        if (defined($vfile)) {
          $dirv = "$ftpdir\\$vfile";
          if ( -r $dirv ) {
            print "  Validations package $dirv present\n";
            system("$copy $dirv $evodir");
          }
          else {
            $errmsg = "File $vfile listed in $rfile missing in $ftpdir!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#-------------------------------------------------------------------------------
#fc      # Check whether all sources have to be recompiled:
#fc      # if there is a sources package and it contains at least one module
#fc      #
#fc      if (defined ($sfile)) {
#fc        print "\n===> Checking source package for full recompilation\n";
#fc        print "===> Gunzipping and untarring $sfile\n";
#fc        system("gunzip $sfile");
#fc        $sbase = $sfile; $sbase =~ s/\.tar\.gz$//;
#fc        system("tar xvf $sbase.tar 2>>$errors");
#fc        system("gzip -f -9 $sbase.tar");
#fc        #
#fc        # Build up list of files in local directory
#fc        opendir (LOCALDIR, ".");
#fc        @files = readdir (LOCALDIR);
#fc        closedir (LOCALDIR);
#fc        # Check whether there are module files (m_*.ff)
#fc        foreach $file (@files) {
#fc          $_ = $file;
#fc          if ( m/^m_[a-z0-9_]+\.ff/ ) {
#fc            $full_compil = "yes";
#fc          }
#fc        }
#fc      }
#-------------------------------------------------------------------------------
        # Process the includes package, if any
        #
        if (defined ($ifile)) {
          print "\n===> Testing includes\n";
          print "===> Gunzipping and untarring $ifile\n";
          system("gunzip $ifile");
          $ibase = $ifile; $ibase =~ s/\.tar\.gz$//;
          system("tar xvf $ibase.tar 2>>$errors");
          system("gzip -f -9 $ibase.tar");
#-------------------------------------------------------------------------------
          # Retrieve the includers, but not if full recompilation is needed
          #
          if ( ! defined ($full_compil)) {
            unlink "epx_get_includers.err";
            system("epx_get_includers");
            if ( -r "epx_get_includers.err" ) {
              $errmsg = "Die epx_evol_start: ERROR in epx_get_includers!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
        }
#-------------------------------------------------------------------------------
        # Process the sources package, if any
        #
        if (defined ($sfile)) {
          print "\n===> Testing sources\n";
          #
          # If full recompilation needed, retrieve all sources
          if (defined ($full_compil)) {
            print "\n===> Retrieving all sources (full recompilation needed)\n";
            if (system ("$copy $srcdir\\*.ff .")) {
              $errmsg = "Die epx_evol_start: ERROR retrieve all sources!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          #
          print "===> Gunzipping and untarring $sfile\n";
          system("gunzip $sfile");
          $sbase = $sfile; $sbase =~ s/\.tar\.gz$//;
          system("tar xvf $sbase.tar 2>>$errors");
          system("gzip -f -9 $sbase.tar");
#-------------------------------------------------------------------------------
          # Verify that a proper history file is contained within the sources
          # package
          #
          $sfileh = $sfile;
          $sfileh =~ s/\.tar\.gz$//;
          $sfileh =~ s/^S_/Cs/;
          $sfileh = "$sfileh\.hist";
          if (! -r $sfileh) {
            $errmsg = "History file $sfileh missing in sources package!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
#-------------------------------------------------------------------------------
          # Remove the epx_ordo.txt file if present, because it
          # contains only the names in the sources package and NOT those
          # of the retrieved includers.
          # A new, complete epx_ordo.txt will be automatically
          # generated when invoking epx_cmp without file names!
          # Same thing in case full recompilation is needed!
          #
          if (defined ($ifile) || defined ($full_compil)) {
            if ( -r "epx_ordo.txt" ) {
              print "\n===> Removing epx_ordo.txt file from sources package.\n";
              unlink "epx_ordo.txt";
            }
          }
#-------------------------------------------------------------------------------
          # Compile the sources
          #
          unlink "epx_cmp.err";
          system("epx_cmp -q -o");
          if ( -r "epx_cmp.err" ) {
            $errmsg = "Die epx_evol_start: compilation ERROR (epx_cmp)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#-------------------------------------------------------------------------------
        # Link and produce an executable module, if necessary. If there are
        # benchmarks but neither sources nor includes in evolution, then
        # no link is performed, but the standard executable is
        # copied locally under the name epx.exe.
        #
        if (defined($sfile) || defined($ifile) || defined($bfile)) {
          #
          # Benchs are necessary because this is not a "manuals-only"
          # evolution!
          #
          if (defined($sfile) || defined($ifile)) {
            print "\n===> Linking to produce a local library (libplex.lib)\n";
            print "\n===> and executable (epx.exe).\n";
            #
            unlink "epx_lk.err";
            if (defined ($full_compil)) {
              # Full recompilation has been done. Build up the local library
              # ex-novo, i.e. by ignoring the standard one (-L switch)
              system("epx_lk -L -o");
            }
            else {
              system("epx_lk -l -o");
            }
            if ( -r "epx_lk.err" ) {
              $errmsg = "Die epx_evol_start: ERROR in epx_lk!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          else {
            print "\n===> Copying standard executable to epx.exe\n";
#fc use copyux otherwise it prompts user: is it a file or directory name!
            system ("$copyux $stdexe epx.exe");
          }
#-------------------------------------------------------------------------------
          # Execute the benchmark test suite
          #
          print "\n===> Executing the benchmark test suite\n";
          #
          # Process the benchmarks package, if any
          #
          if (defined ($bfile)) {
            print "===> Gunzipping and untarring $bfile\n";
            system("gunzip $bfile");
            $bbase = $bfile; $bbase =~ s/\.tar\.gz$//;
            system("tar xvf $bbase.tar 2>>$errors");
            system("gzip -f -9 $bbase.tar");
            #
            # Verify that a proper history file is contained within the
            # benchmarks package
            #
            $bfileh = $bfile;
            $bfileh =~ s/\.tar\.gz$//;
            $bfileh =~ s/^B_/Cb/;
            $bfileh = "$bfileh\.hist";
            if (! -r $bfileh) {
              $errmsg =
              "History file $bfileh missing in benchmarks package!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          #
```

```
          # Run the local tests (if any). Run also the standard
          # benchmark tests, if either the sources or the includes
          # (or both) have changed.
          #
          if (defined($sfile) || defined($ifile)) {
            $local = "";
          }
          else {
            $local = "-l";
          }
          #
          unlink "epx_test_benchmarks.err";
          system("epx_test_benchmarks $local");
          if ( -r "epx_test_benchmarks.err" ) {
            $errmsg = "Die epx_evol_start: ERROR in epx_test_benchmarks!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#-------------------------------------------------------------------------------
        # Update the includes library if necessary
        #
        if (defined($ifile)) {
          print "\n===> Updating the includes library\n";
          if (system ("$copy *.inc $incdir")) {
            $errmsg = "Die epx_evol_start: ERROR updating the includes!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#-------------------------------------------------------------------------------
        # Update the sources library if necessary
        #
        if (defined($sfile)) {
          print "\n===> Updating the sources library\n";
          if (system ("$copy *.ff $srcdir")) {
            $errmsg = "Die epx_evol_start: ERROR updating the sources!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#-------------------------------------------------------------------------------
        # Update the modules (.mod) library if necessary
        #
        if (defined($ifile) || defined($sfile)) {
          #
          # Verify that there are > 0 mod files, to avoid copy error
          #
          $nummod = `ls -l *.mod 2>>$errors | wc -l`;
          chop $nummod;
          $nummod =~ s/[ \t]//g;
          if ($nummod > 0) {
            print "\n===> Updating the modules library\n";
            if (system ("$copy *.mod $moddir")) {
              $errmsg = "Die epx_evol_start: ERROR updating the modules!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
        }
#-------------------------------------------------------------------------------
        # Update the objects (.lib) library if necessary
        #
        if (defined($ifile) || defined($sfile)) {
          print "\n===> Updating the objects library\n";
          if (system ("$copy libplex.lib $libdir")) {
            $errmsg = "Die epx_evol_start: ERROR updating the library!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
#-------------------------------------------------------------------------------
        # Update the executable if necessary
        #
        if (defined($sfile) || defined($ifile)) {
          print "\n===> Updating the executable\n";
          if (system ("$copy epx.exe $stdexe")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the executable!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          #
          # Update also the on-line executable
          #
          print "\n===> Update the on-line executable (see evol_putexe.log)\n";
#fc use copyux because of LOCAL copy!
          system ("$copyux epx.exe europlexus.exe");
          system ("gzip -f -9 europlexus.exe");
#         if (system ("$copy europlexus.exe.gz $wwwexe\\europlexus\.exe\.gz")) {
#           $errmsg =
#           "Die epx_evol_start: ERROR updating the on-line executable!\007\n";
#           print "$errmsg";
#           &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
#         }
          system ("epx_ftp_putexe");
#
          system ("$del europlexus.exe");
          #-----------------------------------------------------------------
          # Generate also the QuickWin executable and update it
          if (! defined ($noqw)) {
            print "\n===> Linking to produce a local library (libplex.lib)\n";
            print "\n===> and QuickWin executable (epxqw.exe).\n";
            #
            unlink "epx_lk.err";
            if (defined ($full_compil)) {
              # Full recompilation has been done. Build up the local library
              # ex-novo, i.e. by ignoring the standard one (-L switch)
              system("epx_lk -L -o -w");
            }
            else {
              system("epx_lk -l -o -w");
            }
            if ( -r "epx_lk.err" ) {
              $errmsg = "Die epx_evol_start: ERROR in epx_lk -w!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            print "\n===> Updating the QuickWin executable\n";
            if (system ("$copy epxqw.exe $stdqwexe")) {
              $errmsg =
              "Die epx_evol_start: ERROR updating the QuickWin executable!\007\n
";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            # Update also the QuickWin on-line executable
            print "\n===> Update the QuickWin on-line executable\n";
            print " (see evol_putexe.log)\n";
#fc use copyux because of LOCAL copy!
            system ("$copyux epxqw.exe europlexusqw.exe");
            system ("gzip -f -9 europlexusqw.exe");
            system ("epx_ftp_putexe -w");
            system ("$del europlexusqw.exe");
          }
          else {
            print "\n===> NOT Updating the QuickWin version (-noqw switch)\n";
          }
          #-----------------------------------------------------------------
          # Generate also the MPI executable and update it
          #
          if (! defined ($nompi)) {
            print "\n===> Updating the MPI executable\n";
            #
            # Remove work directory MPI if already existing
            if (-d "MPI") {
              print "\n  ===> Deleting the previous MPI directory\n";
# use rd instead of rmdir since in some cases rmdir seems to use
# GNU's rmdir instead of DOS rmdir (which is the one I want to use here)
#             system ("rmdir /s/q MPI >>$errors 2>>&1");
              system ("rd /s/q MPI >>$errors 2>>&1");
#
            }
            # Create a work directory MPI and go in it
            print "\n  ===> Creating the MPI directory\n";
            if (system ("mkdir MPI")) {
              $errmsg =
              "Die epx_evol_start: ERROR mkdir MPI!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            print "\n  ===> Changing to the MPI directory\n";
            if (system ("chdir MPI")) {
              $errmsg =
              "Die epx_evol_start: ERROR chdir MPI!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            if (system ("$copy ..\\*.ff .")) {
              $errmsg =
              "Die epx_evol_start: ERROR copying sources for MPI!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            if (defined($ifile)) {
              if (system ("$copy ..\\*.inc .")) {
                $errmsg =
                "Die epx_evol_start: ERROR copying includes for MPI!\007\n";
                print "$errmsg";
                &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
              }
            }
            print "\n===> Compiling sources for MPI version\n";
            system("epx_cmp -q -o -M");
            if ( -r "epx_cmp.err" ) {
              $errmsg = "Die epx_evol_start: MPI compilation ERROR (epx_cmp)!\00
7\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            print "\n===> Linking MPI version\n";
            if (defined ($full_compil)) {
              # Full recompilation has been done. Build up the local library
              # ex-novo, i.e. by ignoring the standard one (-L switch)
              system("epx_lk -L -o -M");
            }
            else {
              system("epx_lk -l -o -M");
            }
            if ( -r "epx_lk.err" ) {
              $errmsg = "Die epx_evol_start: MPI linkage ERROR in epx_lk!\007\n"
;
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            print "\n===> Updating the MPI executable\n";
            if (system ("$copy epx.exe $stdMPIexe")) {
              $errmsg =
              "Die epx_evol_start: ERROR updating the MPI executable!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            # Update also the MPI on-line executable
            print "\n===> Update the MPI on-line executable\n";
            print " (see evol_putexe.log)\n";
#fc use copyux because of LOCAL copy!
            system ("$copyux epx.exe europlexus_mpi.exe");
            system ("gzip -f -9 europlexus_mpi.exe");
            system ("epx_ftp_putexe -M");
            system ("$del europlexus_mpi.exe");
            # Update the modules (.mod) library if necessary
            #
            #
            # Verify that there are > 0 mod files, to avoid copy error
            #
            $nummod = `ls -l *.mod 2>>$errors | wc -l`;
            chop $nummod;
            $nummod =~ s/[ \t]//g;
            if ($nummod > 0) {
              print "\n===> Updating the MPI modules library\n";
              if (system ("$copy *.mod $MPImoddir")) {
                $errmsg = "Die epx_evol_start: ERROR updating the MPI modules!\0
07\n";
                print "$errmsg";
                &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
              }
```

```
            }
            # Update the objects (.lib) library if necessary
            #
            print "\n===> Updating the MPI objects library\n";
            if (system ("$copy libplex_MPI.lib $libdir")) {
              $errmsg = "Die epx_evol_start: ERROR updating the MPI library!\007
\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            # Return to parent directory
            print "\n  ===> Changing back to parent directory\n";
            if (system ("chdir ..")) {
              $errmsg =
              "Die epx_evol_start: ERROR chdir ..!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            # Clean up (remove entire temporary directory MPI)
            print "\n  ===> Removing the MPI work directory\n";
            system ("rd /s/q MPI >>$errors 2>>&1");
            #
          }
          else {
            print "\n===> NOT Updating the MPI executable (-nompi switch)\n";
          }
          #---------------------------------------------------------------------
          # Generate also the -check executable and update it
          #
          if (! defined ($nocheck)) {
            print "\n===> Updating the -check executable\n";
            #
            # Remove work directory CHECK if already existing
            if (-d "CHECK") {
              print "\n  ===> Deleting the previous CHECK directory\n";
# use rd instead of rmdir since in some cases rmdir seems to use
# GNU's rmdir instead of DOS rmdir (which is the one I want to use here)
#             system ("rmdir /s/q CHECK >>$errors 2>>&1");
              system ("rd /s/q CHECK >>$errors 2>>&1");
#
            }
            # Create a work directory CHECK and go in it
            print "\n  ===> Creating the CHECK directory\n";
            if (system ("mkdir CHECK")) {
              $errmsg =
              "Die epx_evol_start: ERROR mkdir CHECK!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            print "\n  ===> Changing to the CHECK directory\n";
            if (system ("chdir CHECK")) {
              $errmsg =
              "Die epx_evol_start: ERROR chdir CHECK!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            if (system ("$copy ..\\*.ff .")) {
              $errmsg =
              "Die epx_evol_start: ERROR copying sources for CHECK!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            if (defined($ifile)) {
              if (system ("$copy ..\\*.inc .")) {
                $errmsg =
                "Die epx_evol_start: ERROR copying includes for CHECK!\007\n";
                print "$errmsg";
                &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
              }
            }
            print "\n===> Compiling sources for CHECK version\n";
            system("epx_cmp -q -o -c");
            if ( -r "epx_cmp.err" ) {
              $errmsg = "Die epx_evol_start: CHECK compilation ERROR (epx_cmp)!\
007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            print "\n===> Linking CHECK version\n";
            if (defined ($full_compil)) {
              # Full recompilation has been done. Build up the local library
              # ex-novo, i.e. by ignoring the standard one (-L switch)
              system("epx_lk -L -o -c");
            }
            else {
              system("epx_lk -l -o -c");
            }
            if ( -r "epx_lk.err" ) {
              $errmsg = "Die epx_evol_start: CHECK linkage ERROR in epx_lk!\007\
n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            print "\n===> Updating the CHECK executable\n";
            if (system ("$copy epx.exe $stdCHECKexe")) {
              $errmsg =
              "Die epx_evol_start: ERROR updating the CHECK executable!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            # Update also the -check on-line executable
            print "\n===> Update the -check on-line executable\n";
            print " (see evol_putexe.log)\n";
#fc use copyux because of LOCAL copy!
            system ("$copyux epx.exe europlexus_check.exe");
            system ("gzip -f -9 europlexus_check.exe");
            system ("epx_ftp_putexe -c");
            system ("$del europlexus_check.exe");
            # Update the objects (.lib) library if necessary
            #
            print "\n===> Updating the CHECK objects library\n";
            if (system ("$copy libplex_check.lib $libdir")) {
              $errmsg = "Die epx_evol_start: ERROR updating the CHECK library!\0
07\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            # Return to parent directory
            print "\n  ===> Changing back to parent directory\n";
            if (system ("chdir ..")) {
              $errmsg =
              "Die epx_evol_start: ERROR chdir ..!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
            # Clean up (remove entire temporary directory CHECK)
            print "\n  ===> Removing the CHECK work directory\n";
            system ("rd /s/q CHECK >>$errors 2>>&1");
            #
          }
          else {
            print "\n===> NOT Updating the -check executable (-nocheck switch)\n
";
          }
        }
#---------------------------------------------------------------------------
        # Update the benchmarks library if necessary
        #
        if (defined($sfile) || defined($ifile) || defined($bfile)) {
          print "\n===> Updating the benchmarks\n";
          #
          # Input data file is mandatory
          #
          if (system ("$copy *.epx $bendir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the benchmarks(1)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          #
          # Mesh file is optional
          # Verify that there are > 0 msh files, to avoid copy error
          #
          $nummsh = `ls -l *.msh 2>>$errors | wc -l`;
          chop $nummsh;
          $nummsh =~ s/[ \t]//g;
          if ($nummsh > 0) {
            if (system ("$copy *.msh $bendir")) {
              $errmsg =
              "Die epx_evol_start: ERROR updating the benchmarks(2)!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          #
          # Zip file (bm_*.zip) is optional
          # Verify that there are > 0 zip files, to avoid copy error
          #
          $numzip = `ls -l bm_*.zip 2>>$errors | wc -l`;
          chop $numzip;
          $numzip =~ s/[ \t]//g;
          if ($numzip > 0) {
            if (system ("$copy bm_*.zip $bendir")) {
              $errmsg =
              "Die epx_evol_start: ERROR updating the benchmarks(3)!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          #
          # Listing file is mandatory
          #
          if (system ("$copy *.listing $bendir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the benchmarks(4)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          #
          # PostScript file is optional
          # Verify that there are > 0 ps files, to avoid copy error
          #
          $numps = `ls -l *.ps 2>>$errors | wc -l`;
          chop $numps;
          $numps =~ s/[ \t]//g;
          if ($numps > 0) {
            if (system ("$copy *.ps $bendir")) {
              $errmsg =
              "Die epx_evol_start: ERROR updating the benchmarks(5)!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          #
          # BMP file(s) are optional
          # Verify that there are > 0 bmp files, to avoid copy error
          #
          $numbmp = `ls -l *.bmp 2>>$errors | wc -l`;
          chop $numbmp;
          $numbmp =~ s/[ \t]//g;
          if ($numbmp > 0) {
            if (system ("$copy *.bmp $bendir")) {
              $errmsg =
              "Die epx_evol_start: ERROR updating the benchmarks(6)!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          #
          # AVI file(s) are optional
          # Verify that there are > 0 avi files, to avoid copy error
          #
          $numavi = `ls -l *.avi 2>>$errors | wc -l`;
          chop $numavi;
          $numavi =~ s/[ \t]//g;
          if ($numavi > 0) {
            if (system ("$copy *.avi $bendir")) {
              $errmsg =
              "Die epx_evol_start: ERROR updating the benchmarks(7)!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          #
          # PVD file(s) are optional
          # Verify that there are > 0 pvd files, to avoid copy error
```

```
          #
          $numpvd = `ls -l *.pvd 2>>$errors | wc -l`;
          chop $numpvd;
          $numpvd =~ s/[ \t]//g;
          if ($numpvd > 0) {
            if (system ("$copy *.pvd $bendir")) {
              $errmsg =
              "Die epx_evol_start: ERROR updating the benchmarks(8)!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          #
          # VTU file(s) are optional
          # Verify that there are > 0 vtu files, to avoid copy error
          #
          $numvtu = `ls -l *.vtu 2>>$errors | wc -l`;
          chop $numvtu;
          $numvtu =~ s/[ \t]//g;
          if ($numvtu > 0) {
            if (system ("$copy *.vtu $bendir")) {
              $errmsg =
              "Die epx_evol_start: ERROR updating the benchmarks(9)!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
        }
#-------------------------------------------------------------------------------
        # Process the manuals package, if any
        #
        if (defined($mfile)) {
          print "\n===> Testing manuals\n";
          print "===> Gunzipping and untarring $mfile\n";
          system("gunzip $mfile");
          $mbase = $mfile; $mbase =~ s/\.tar\.gz$//;
          system("tar xvf $mbase.tar 2>>$errors");
          system("gzip -f -9 $mbase.tar");
          #
          # Verify that a proper history file is contained within the
          # manuals package
          #
          $mfileh = $mfile;
          $mfileh =~ s/\.tar\.gz$//;
          $mfileh =~ s/^M_/Cm/;
          $mfileh = "$mfileh\.hist";
          if (! -r $mfileh) {
            $errmsg = "History file $mfileh missing in manuals package!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
#-------------------------------------------------------------------------------
          # Test the manuals
          #
          unlink "epx_test_manuals.err";
          system("epx_test_manuals");
          if ( -r "epx_test_manuals.err" ) {
            $errmsg = "Die epx_evol_start: ERROR in epx_test_manuals!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
#-------------------------------------------------------------------------------
          # Update the manuals library
          #
          print "\n===> Updating the manuals\n";
          if (system ("$copy *.ttx $mandir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the manuals(1)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  LaTeX sources updated.\n";
#
          if (system ("$copy *.tex $manfildir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the filtered manuals(1)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  LaTeX filtered sources updated.\n";
#
          if (system ("$copy manual.dvi $mandir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the manuals(2)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  DVI version updated.\n";
#
          if (system ("$copy manual.ps $mandir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the manuals(3)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  PostScript version updated.\n";
#
          if (system ("$copy manual.pdf $mandir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the manuals(4)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  PDF version updated.\n";
#
          if (system ("$copy manual.html $mandir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the manuals(5)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  HTML (Tth) version updated.\n";
#
          if (system ("$copy manual_h.html $mandir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the manuals(6)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  HTML (Hevea/Hacha) monolithic version updated.\n";
#
          if (system ("$copy hacha\\*.* $hacdir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the manuals(7)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  HTML (Hevea/Hacha) split version updated.\n";
#
          #
          # Update also the on-line manual (PDF + Hevea/Hacha split versions)
          #
#         if (system ("$copy manual.pdf $wwwman")) {
#           $errmsg =
#           "Die epx_evol_start: ERROR updating on-line PDF manual!\007\n";
#           print "$errmsg";
#           &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
#         }
          system ("epx_ftp_putman");
#
          print "  PDF on-line version update (see evol_putman.log).\n";
#
#         if (system ("$copy hacha\\*.* $wwwman")) {
#           $errmsg =
#           "Die epx_evol_start: ERROR updating on-line HTML manual!\007\n";
#           print "$errmsg";
#           &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
#         }
          print "  HTML(Hacha) on-line version update (see evol_putman.log).\n";
        }
#-------------------------------------------------------------------------------
        # Process the validations package, if any
        #
        if (defined($vfile)) {
          print "\n===> Testing validations\n";
          print "===> Gunzipping and untarring $vfile\n";
          system("gunzip $vfile");
          $vbase = $vfile; $vbase =~ s/\.tar\.gz$//;
          system("tar xvf $vbase.tar 2>>$errors");
          system("gzip -f -9 $vbase.tar");
          #
          # Verify that a proper history file is contained within the
          # validations package
          #
          $vfileh = $vfile;
          $vfileh =~ s/\.tar\.gz$//;
          $vfileh =~ s/^V_/Cv/;
          $vfileh = "$vfileh\.hist";
          if (! -r $vfileh) {
            $errmsg = "History file $vfileh missing in validat. package!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
#-------------------------------------------------------------------------------
          # Update the validations library
          #
          print "\n===> Updating the validations\n";
          if (system ("$copy vl_*.vld $valdir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the validations(1)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  Validation description files (vl_*.vld) updated.\n";
#
          if (system ("$copy vl_*.zip $valdir")) {
            $errmsg =
            "Die epx_evol_start: ERROR updating the validations(2)!\007\n";
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
          print "  Validation package files (vl_*.zip) updated.\n";
        }
#-------------------------------------------------------------------------------
        # Process the history file(s), if any, and update the histories
        #
        if (defined($sfileh) || defined($bfileh) || defined($mfileh) ||
            defined($vfileh)) {
          print "\n===> Processing the history file(s)\n";
          unlink "epx_evol_histories.err";
          if (defined($sfileh)) {
            print "  Processing the sources/includes history file(s)\n";
            system("epx_evol_histories $sfileh");
            if ( -r "epx_evol_histories.err" ) {
              $errmsg =
              "Die epx_evol_start: ERROR in epx_evol_histories!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          if (defined($bfileh)) {
            print "  Processing the benchmarks history file(s)\n";
            system("epx_evol_histories $bfileh");
            if ( -r "epx_evol_histories.err" ) {
              $errmsg =
              "Die epx_evol_start: ERROR in epx_evol_histories!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          if (defined($mfileh)) {
            print "  Processing the manuals history file(s)\n";
            system("epx_evol_histories $mfileh");
            if ( -r "epx_evol_histories.err" ) {
              $errmsg =
              "Die epx_evol_start: ERROR in epx_evol_histories!\007\n";
              print "$errmsg";
              &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
            }
          }
          if (defined($vfileh)) {
            print "  Processing the validations history file(s)\n";
            system("epx_evol_histories $vfileh");
            if ( -r "epx_evol_histories.err" ) {
              $errmsg =
              "Die epx_evol_start: ERROR in epx_evol_histories!\007\n";
```

```
            print "$errmsg";
            &ERRMAIL ($logfile, $num, $evolog); die "$errmsg";
          }
        }
        print "Histories updating has successfully terminated.\n";
      }
#-------------------------------------------------------------------------------
      # Copy the resume file, the trace file, and the evolution package
      # files (if any) to the backup directory, then delete them from
      # the FTP directory.
      #
      system("$copy $dirr $bkpdir"); unlink $dirr;
      system("$copy $dirt $bkpdir"); unlink $dirt;
      if (defined($ifile)) {system("$copy $diri $bkpdir"); unlink $diri};
      if (defined($sfile)) {system("$copy $dirs $bkpdir"); unlink $dirs};
      if (defined($bfile)) {system("$copy $dirb $bkpdir"); unlink $dirb};
      if (defined($mfile)) {system("$copy $dirm $bkpdir"); unlink $dirm};
      if (defined($vfile)) {system("$copy $dirv $bkpdir"); unlink $dirv};
#-------------------------------------------------------------------------------
      # Reset the variables that are tested through "defined",
      # in case another evolution has to be performed
      #
      undef $ifile; undef $sfile; undef $bfile; undef $mfile; undef $vfile;
      undef $sfileh; undef $bfileh; undef $mfileh; undef $vfileh;
      undef $ifileunpacked;
#-------------------------------------------------------------------------------
      # The evolution is terminated:
      #   - increment the evolution index;
      #   - clean up the files in $evodir (.);
      #
      print "\n===> EVOLUTION N. $num SUCCESSFUL!\n";
      print "===> Incrementing the evolution index\n";
      #
      system ("echo $num >$evonumfile");
      $evonum = `cat $evonumfile`;
      chop $evonum;
      print "\n===> The current evolution index is  : \# $evonum\n";
      #
      $numok = $evonum + 1;
      print "===> The next evolution index must be: \# $numok\n";
      #
      print "\n===> Cleaning up the evolution directory\n";
      system("$del *.gz"); system("$del *.hist");          # Packages
      system("$del *.ff"); system("$del *.f");             # Sources
      system("$del *.inc");                                # Includes
      system("$del *.obj"); system("$del *.err");          # Compilation
      system("$del *.mod");
      system("$del *.exe"); system("$del *.lib");          # Link
      system("$del *.pdb");
      system("$del *.eco"); system("$del *.std");          # Run/benchs
      system("$del *.epx"); system("$del *.msh");
      system("$del *.listing"); system("$del *.ps");
      system("$del bm_*.log"); system("$del bm_*.zip");
      system("$del bm_*.plog"); system("$del bm_*.pin");
      system("$del bm_*.lks");
      system("$del bm_listing.lst");
      system("$del *.bmp"); system("$del *.ide");
      system("$del *.sau");
      system("$del vl_*.vld"); system("$del vl_*.zip");    # Validations
      system("$del bm_*.ali");                             # Outputs
      system("$del bm_*.alt");
      system("$del bm_*.inp");
      system("$del bm_*.k20");
      system("$del bm_*.k200"); system("$del bm_*.k2000");
      system("$del bm_*.mtv");
      system("$del *.pun");
      system("$del bm_*.puk");
      system("$del bm_*.mgr");
      system("$del bm_*.txt");
      system("$del bm_*.avi");
      system("$del bm_*.pvd");
      system("$del bm_*.vtu");
      system("$del bm_*.str");
      system("$del bm_*.savr");
      system("$del bm_*.tpl");
      system("$del bm_*.unv");
      system("$del bm_*.xpl");
      system("$del bm_*.poc");
      system("$del bm_stderr.lst");
      system("$del epx_ordo.txt");
      system("$del epx*.exp");
      system("$del *.u18");
      system("$del *.dat");
      system("$del *.inp"); system("$del *.mif");
      system("$del *.p10"); system("$del *.pov");
      system("$del fort.*"); system("$del *.k");
      system("$del *.ttx"); system("$del *.aux");          # Manuals
      system("$del *.tex");
      system("$del gbnotice.*"); system("$del manual.*");
      system("$del manual_h.*");
      system("$del epx_pass_*.log");
      system("$del tth.log");
      if (-d "hacha") {
#       system("$del hacha\\*.*");
#       system("$rmdir hacha");
#
# use rd instead of rmdir since in some cases rmdir seems to use
# GNU's rmdir instead of DOS rmdir (which is the one I want to use here)
#       system("rmdir /s/q hacha >>$errors 2>>&1");
        system("rd /s/q hacha >>$errors 2>>&1");
#
      }
      system("$del _*.*");                                 # Temporary
#
# Reset full compilation flag, in case of multiple evolutions !
      if (defined ($full_compil)) {
        undef $full_compil;
      }
#-------------------------------------------------------------------------------
      # LAST OPERATION: reset the evolution lock.
      #
      print "\n===> Removing the evolution lock\n";
      unlink "$evolck";
#-------------------------------------------------------------------------------
# Report user and system CPU times for this process and for its children
#
      ($userb, $systemb, $cuserb, $csystemb) = times;
      $user    = $userb    - $usera;
      $system  = $systemb  - $systema;
      $cuser   = $cuserb   - $cusera;
      $csystem = $csystemb - $csystema;
#
      print "\nThis process used the following CPU times:\n";
      print "User   : $user\n";
      print "System : $system\n";
      print "\nThe children of this process used the following CPU times:\n";
      print "Cuser  : $cuser\n";
      print "Csystem: $csystem\n\n";
#-------------------------------------------------------------------------------
# Get and print current date and time
#
      $time = time;
      $gmttime = time2str($time);
      $loctime = time2iso($time);
      print "===> JRC EUROPLEXUS evolution \# $lognum ended   on: $gmttime\n";
      print "===> Local date/time: $loctime\n";
#-------------------------------------------------------------------------------
      # Redirect STDOUT and STDERR to $logfile
      #
      close STDOUT;
      close STDERR;
      open STDOUT, ">>$logfile";
      open STDERR, ">>$logfile";
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
      $oldfh = select(STDOUT); $| = 1; select($oldfh);
      $oldfh = select(STDERR); $| = 1; select($oldfh);
#
      print "  Redirecting STDOUT and STDERR to $logfile.\n";
      print "\n===> EVOLUTION N. $num SUCCESSFUL at $loctime!\n";
#-------------------------------------------------------------------------------
      # Mail evolution log file to project partners
      #
      print "\n===> Mailing log file $evolog to partners.\n\n";
      unlink "epx_mail.err";
      $stat = "OK";
      system("epx_mail $stat $num $evolog");
      if ( -r "epx_mail.err" ) {
        die "Die epx_evol_start: ERROR in epx_mail!\007\n";
      }
#-------------------------------------------------------------------------------
      # Move evolution log file to trace directory
      #
      print "\n===> Moving evolution log file $evolog\n";
      print "      to $trcdir.\n";
      system("$copy $evolog $trcdir"); unlink $evolog;
    }
    else {
      print "The evolution index is out of sequence: skip this evolution!\n";
    }
  }
}
#-------------------------------------------------------------------------------
# Report user and system CPU times for this process and for its children
#
($user1, $system1, $cuser1, $csystem1) = times;
#
$user    = $user1    - $user0;
$system  = $system1  - $system0;
$cuser   = $cuser1   - $cuser0;
$csystem = $csystem1 - $csystem0;
#
print "\nThis process used the following CPU times:\n";
print "User   : $user\n";
print "System : $system\n";
print "\nThe children of this process used the following CPU times:\n";
print "Cuser  : $cuser\n";
print "Csystem: $csystem\n\n";
#-------------------------------------------------------------------------------
# Get and print current date and time
#
$time = time;                       # Machine time (seconds since epoch)
$gmttime = time2str($time);         # Universal (GMT) date and time
$loctime = time2iso($time);         # This requires function time2iso to be
                                    # exported in module HTTP\Date.pm (is not
                                    # exported by default!
print "===> EUROPLEXUS evolution(s) at JRC ended   on: $gmttime\n";
print "===> Local date/time: $loctime\n";
#-------------------------------------------------------------------------------
# Copy evol.log and evol_*.log files to trace directory
#
close STDOUT;
close STDERR;
system("$copy $logfile $trcdir");
system("$copy $ftplogfile $trcdir");
system("$copy $exelogfile $trcdir");
system("$copy $manlogfile $trcdir");
#
exit;


#-------------------------------------------------------------------------------
sub ERRMAIL {
  local ($logfile, $num, $evolog) = @_;
#
  close STDOUT;
  close STDERR;
  open STDOUT, ">>$logfile";
  open STDERR, ">>$logfile";
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
  $oldfh = select(STDOUT); $| = 1; select($oldfh);
  $oldfh = select(STDERR); $| = 1; select($oldfh);
#
  print "\n===> Mailing log file $evolog to partners.\n\n";
  unlink "epx_mail.err";
  $stat = "FAILED!!!";
  system("epx_mail $stat $num $evolog");
  if ( -r "epx_mail.err" ) {
    die "Die epx_evol_start: ERROR in epx_mail!\007\n";
  }
}
#-------------------------------------------------------------------------------
```

epx_evol_test.pl

```
#-------------------------------------------------------------------------------
# Start evolution of EUROPLEXUS files. All evolution packages present in the
# FTP directory are treated. If more than one evolution package set is
# present, they are applied in order.
#-------------------------------------------------------------------------------
# Modules needed
#
use HTTP::Date;
#-------------------------------------------------------------------------------
# Redirect STDOUT and STDERR to evol.log (Fixed path!!!!)
#
$logfile = "C:\\EUROPLEXUS\\Fromcentral\\evol.log";
open STDOUT, ">>$logfile";
open STDERR, ">>$logfile";
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Get and print current date and time
#
$time = time;                       # Machine time (seconds since epoch)
$gmttime = time2str($time);         # Universal (GMT) date and time
$loctime = time2iso($time);         # This requires function time2iso to be
                                    # exported in module HTTP\Date.pm (is not
                                    # exported by default!
print "\n\n";
print "========================================";
print "========================================\n";
print "===> EUROPLEXUS evolution(s) at JRC started on: $gmttime\n";
print "===> Local date/time: $loctime\n";
#-------------------------------------------------------------------------------
# Default values
#
#$ftpdir = "C:\\IIS\\FTP\\EUROPLEXUS"; # Directory receiving FTP files
$ftpdir = "C:\\EUROPLEXUS\\FTP"; # Directory receiving FTP files
#
$wwwdir = "C:\\WWW\\EUROPLEXUS"; # Directory where to put on-line files
$wwwman = "$wwwdir\\MANUAL";     # Directory where to put on-line manual
$wwwexe = "$wwwdir\\EXE";        # Directory where to put on-line executable
#
#fc 02 May 2005
$copyux = "cp -p -f";
$copy = "xcopy /Q /R /Y /I";
#fc 02 May 2005
$del  = "rm -f";
#-------------------------------------------------------------------------------
$gnudir = $ENV{'GNUDIR'};
if ( ! defined $gnudir) {
  die "The GNUDIR environment variable is undefined!\007\n";
}
$rmdir = "$gnudir\\rmdir.exe";   # Full path needed! Else uses MS-DOS's rmdir!
$errors = "C:\\EUROPLEXUS\\Fromcentral\\errors.txt";
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 0 or 1 or 2 or 3 or 4
#
if ($#ARGV > 3) {  # Index of last argument must be = -1 or 0 or 1 or 2 or 3
  die "Usage: epx_evol_test [-p] [-noqw] [-nompi] [-nocheck]\007\n";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#       -p       patch (do not get evolution files by ftp)
#       -noqw    do not update the QuickWin executable
#       -nompi   do not update the MPI executable
#       -nocheck do not update the -check executable
#
#print "Index of last arg is: $#ARGV.\n";
$nar = $#ARGV + 1;
#print "No. of args is: $nar.\n";
if ($#ARGV >= 0) { # There is at least one argument
  while ($ARGV[0] =~ /^-/) {
    $_ = shift;
    $nar = $nar - 1;
    if (/^-p(.*)/) {
      $patch = "yes";
      print "Command line switch: patch\n";
    }
    elsif (/^-noqw(.*)/) {
      $noqw = "yes";
      print "Command line switch: noqw\n";
    }
    elsif (/^-nompi(.*)/) {
      $nompi = "yes";
      print "Command line switch: nompi\n";
    }
    elsif (/^-nocheck(.*)/) {
      $nocheck = "yes";
      print "Command line switch: nocheck\n";
    }
    else {
      die "ERROR: unknown switch: $_\n";
    }
  }
}
#print "Index of last arg is: $#ARGV.\n";
#print "No. of args is: $nar.\n";
#-------------------------------------------------------------------------------
# Get and print current date and time
#
$time = time;                       # Machine time (seconds since epoch)
$gmttime = time2str($time);         # Universal (GMT) date and time
$loctime = time2iso($time);         # This requires function time2iso to be
                                    # exported in module HTTP\Date.pm (is not
                                    # exported by default!
print "===> EUROPLEXUS evolution(s) at JRC ended   on: $gmttime\n";
print "===> Local date/time: $loctime\n";
#-------------------------------------------------------------------------------
close STDOUT;
close STDERR;
#
exit;
#-------------------------------------------------------------------------------
```

epx_fil.pl

```
#-------------------------------------------------------------------------------
# Run epx_filter on a whole set of text (ASCII) files
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$in = "ff";
$out = "f";
$b = " ";
#-------------------------------------------------------------------------------
# Check number of arguments, must be >= 1
#
if ($#ARGV < 0) {                       # Index of last argument must be >= 0
  die "Usage: epx_fil [-i<ext>] [-o<ext>] [-c[<char>]] [-p<pwd> ...]
       file(s)\007\n";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#       -i<ext>  : input  files extension, default is ff
#       -o<ext>  : output files extension, default is f
#       -c       : passed unchanged to epx_filter
#       -c<char> : passed unchanged to epx_filter
#       -p<pwd>  : password. May be repeated, and in that case
#                  the various passwords are concatenated
#
while ($ARGV[0] =~ /^-/) {
  $_ = shift;
  if (/^-i(.*)/) {
    $in = ($1 ? $1 : shift);
#    printf "Command line switch: in=$in\n";
  }
  elsif (/^-o(.*)/) {
    $out = ($1 ? $1 : shift);
#    printf "Command line switch: out=$out\n";
  }
  elsif (/^-p(.*)/) {
    $passr = ($1 ? $1 : shift);
#    printf "Command line switch: pass=$passr\n";
    if (defined $pass) {
      $pass = "$pass$b$passr";
    }
    else {
      $pass = $passr;
    }
  }
#-------------------------------------------------------------------------------
# We first chech for an option -c without <char>
#
  elsif ($_ eq "-c") {
    $c = "-c";
    printf "Command line switch: cr=$c\n";
  }
#-------------------------------------------------------------------------------
# Then we chech for an option -c with <char>
#
  elsif (/^-c(.*)/) {
    $cr = ($1 ? $1 : shift);
    $c = "-c$cr";
    printf "Command line switch: cr=$c\n";
  }
  else {
    die "ERROR: unknown switch: $_\n";
  }
}
#-------------------------------------------------------------------------------
# All remainaing arguments are file names: do the globbing
#
@FileNames = glob(join($",@ARGV));
#-------------------------------------------------------------------------------
# Loop on file(s) to be compiled
#
FILE: foreach $file (@FileNames) {

  $base = $file;
  $base =~ s/\.$in$//;

# print("$file [$pass] --> $base.$out : exit code = ");
# print("Command: epx_filter $c $pass <$file >$base.$out");

  if (! system("epx_filter $c $pass <$file >$base.$out")) {
    print "An error occurred while filtering file $file!\007\n";
    system ("ren $base.$out $base.err");
    next FILE;
  }
}
#
exit;
```

epx_filbat.bat

```
@echo off
echo Filtering %1.ff
%UTILDIR%\epx_filter.exe WIN OGL W32 SPLIB MFFT MKL %2 %3 <%1.ff >%1.f
echo Filtering done
```

epx_filbat_noogl.bat

```
@echo off
echo Filtering with key WIN (no OGL) %1.ff
%UTILDIR%\epx_filter.exe WIN SPLIB %2 %3 <%1.ff >%1.f
echo Filtering done
```

epx_filbat_quickwin.bat

```
@echo off
echo Filtering %1.ff
%UTILDIR%\epx_filter.exe WIN OGL W32 SPLIB QWIN %2 %3 <%1.ff >%1.f
echo Filtering done
```

epx_filter.f

```
      PROGRAM epx_filter
!
! New version (fc, 7/10/2004) allowing single-level nesting of conditionals
!
! New version (fc, 7/10/2008) : Correct bug in COMPARE_PASSWORDS, now
!                               one may effectively have up to NBRAN
!                               conditional branches CELIF (as was
!                               initially foreseen), not just one!
!                               Correct bug in VERIFY_PASSWORDS : in case
!                               of error the STOP code is "0" and not "0 1"
!                               (which was returned as 1).
! New version (hb, juillet_09): Syntax A) et C) donne les memes resultats
!                               que le syntax B)
!                               le fichier filtré conserve la meme
!                               longueur de le fichier original
!                               C'est tres utile pour un debogage.
!May 2011                      : ajout du flag "%SALOME"
!July 2011                     : filtrage des chaines '![' et ']!' destinee a ai
der
!                                la mise en donnees dans SALOME
!
!
!
!
c Filters text files containing the following 3 types of construct:
c
c       Syntax A)            |    Syntax B)             |    Syntax C)
c                            |                          |
c     CIF mot1 [mot2 ...]    |  *IF mot1 [mot2 ...]     |  CIFNOT mot1 [mot2 ...]
c         [text1]            |      [text1]             |      [text1]
c    [CELIF mot3 [mot4 ...]] | [CELIF mot3 [mot4 ...]] | [CELSE]
c         [text2]            |      [text2]             |      [text2]
c    [CELSE]                 | [CELSE]                  |  CENDIF
c         [text3]            |      [text3]             |
c     CENDIF                 |  CENDIF                  |
c
c Components in [ ] are optional.
c
c The mot1, mot2 etc are passwords given on the command line by the user.
c
c The concatenation of two or more passwords corresponds to a
c logical .OR. between them. Thus CIF mot1 mot2 means IF (mot1 .OR. mot2)
c and CIFNOT mot1 mot2 means IF (.NOT. (mot1 .OR. mot2)).
c
c The CELSE and CENDIF passwords may be optionally followed by a
c comment on the same line. Comments must be separated by at least
c one blank from the preceding keyword. For example:
c CIF toto        <- 'toto' is a password here
c ELSE non-toto   <- 'non-toto' is a comment
c CENDIF toto     <- 'toto' is a comment
c
c Usage: epx_filter [-c<character>] [mot1 [mot2 ...]] <input >output
c
c
c
c For syntaxes A) and C):
c ======================
c
c       -c<character>   This option, if present, causes the use of
c                       the given <character> in place of C in the
c                       output transcription of keywords, and of
c                       the built-in message *** ACCES INTERDIT ***,
c                       that is automatically output in place
c                       of the non-activated text blocks.
c                       Example: '-c%' would give in output %IF,
c                       %ELIF, %ELSE, %ENDIF and %** ACCES INTERDIT ***.
c                       This example may be used for LaTeX documents, that
c                       use % as comment character.
c
c       -c              If no character (i.e. a blank) is specified, then
c                       neither the keyword lines nor the built-in
c                       message are transcribed, so the output file
c                       will contain just the activated text.
c
c For syntax B):
c =============
c
c The non-active branches, if any, are transcribed preceded by the chosen
c (or default) comment character. They remain therefore visible (but inactive)
c in the filtered text.
c
c       -c<character>   This option, if present, causes the use of
c                       the given <character> in place of C in the
c                       output transcription of keywords, and of
c                       (commented) inactive branches.
c
c       -c              NO EFFECT with this syntax
c
!!Stop HB(juillet_09):    Cette partie est obsolete

c Note that:
c =========
c
c    - Keywords CIF, CELIF etc. MUST start in column 1.
c
c    - Keywords are NOT case sensitive, i.e. CIF, cif or CiF are the same.
c
c    - Passwords mot1, mot2 etc. are NOT case sensitive, and may not
c      contain blanks.
c
c    - To avoid ambiguous cases, the filter performs the following tests
c      while filtering the text file:
c
c        Tests on the command line syntax:
c
c          1. The user may not specify twice the same password (or equivalent
c             passwords that differ only by letter case) on the command line.
c
c        Tests on the input text file contents:
c
c          2. Passwords in each branch of a conditional must differ from
c             any passwords in other branches of the same conditional.
c          3. If a user specifies more than one password, all passwords must
c             belong to the same branch of a conditional.
c          4. The CIFNOT syntax does not admit CELIF branches.
c
c    - The program returns 1 if everything was OK, else it returns 0
c
CIF WIN
c ATTENTION! LA LIGNE SUIVANTE DOIT ETRE COMMENTEE SOUS AIX
      USE dflib ! For GETARG, NARGS
CENDIF
c
      IMPLICIT NONE
c
      INTEGER(4), PARAMETER :: pwdlen = 16 ! Max. length of a password
      INTEGER(4), PARAMETER :: npassw = 9  ! Max. number of passwords
c                                          ! given in command line
      INTEGER(4), PARAMETER :: crdlen = 132! Max. length of a text 'card'
c
      INTEGER(4) :: nar, npa = 0, iline = 0
      INTEGER(4) :: i
CIF WIN
c ATTENTION! LA LIGNE SUIVANTE DOIT ETRE COMMENTEE SOUS AIX
      INTEGER(2) :: pos, stat
CELSE
C ATTENTION! LA LIGNE SUIVANTE DOIT ETRE ACTIVEE SOUS AIX
C     INTEGER(4) :: pos, stat, iargc
CENDIF
      CHARACTER (LEN=pwdlen) :: pwd
      CHARACTER (LEN=pwdlen), DIMENSION(npassw) :: passw  ! Passwords in cmdline
      CHARACTER (LEN=1) :: first = 'C'
      LOGICAL(4) :: echo = .TRUE., inside = .FALSE.
      LOGICAL(4) :: active = .FALSE., found = .FALSE.
      LOGICAL(4) :: star = .FALSE., negative = .FALSE.
      LOGICAL(4) :: elsed = .FALSE.
!
      LOGICAL(4) :: inside2 = .FALSE.
      LOGICAL(4) :: active2 = .FALSE., found2 = .FALSE.
      LOGICAL(4) :: star2 = .FALSE., negative2 = .FALSE.
      LOGICAL(4) :: elsed2 = .FALSE.
c
c echo   : TRUE  if keywords and built-in message are to be transcribed,
c                preceded by the comment character
c          FALSE otherwise
c
c inside : TRUE  if we are inside a CIF or *IF or CIFNOT construct
c          FALSE otherwise
c
c active : TRUE  if we are in the active branch of the construct
c          FALSE otherwise
c
c star   : TRUE  if we are in an *IF construct
c          FALSE otherwise
c
c negative: TRUE  if we are in a CIFNOT construct
c           FALSE otherwise
!
! xxx2 : same meaning as xxx but for the nested conditional
!
! We assume throughout that inside2 implies inside !!!!!!!!!!!!!!!!!!!!
c
      CHARACTER (LEN=crdlen) :: card
      CHARACTER (LEN=4) :: cif    = "CIF "
      CHARACTER (LEN=4) :: sif    = "*IF "
      CHARACTER (LEN=7) :: cifnot = "CIFNOT "
      CHARACTER (LEN=6) :: celif  = "CELIF "
      CHARACTER (LEN=6) :: celse  = "CELSE "
      CHARACTER (LEN=7) :: cendif = "CENDIF "
      CHARACTER (LEN=7) :: c7
      CHARACTER (LEN=7) :: csalome= "%SALOME"
c
      INTEGER(4), PARAMETER :: npasst = 9  ! Max. number of passwords
c                                          ! in each line of text file
      INTEGER(4), PARAMETER :: nbran  = 10 ! Max. n. of branches in a CIF
c                                          ! construct
      INTEGER(4), DIMENSION(nbran) :: npt = 0
      INTEGER(4) :: nbr = 0
      CHARACTER (LEN=pwdlen),
     &           DIMENSION(npasst, nbran) :: passt  ! Passwords in each
c                                                    branch of txtfile
! same as above but for the nested conditional
      INTEGER(4), DIMENSION(nbran) :: npt2 = 0
      INTEGER(4) :: nbr2 = 0
      CHARACTER (LEN=pwdlen),
     &           DIMENSION(npasst, nbran) :: passt2 ! Passwords in each
c                                                    branch of txtfile
!
      CHARACTER (LEN=21) :: msg = "** ACCES INTERDIT ***"
c
      CHARACTER (LEN=80) :: blabla
      INTEGER (4) :: ioerror
c
      INTEGER, PARAMETER :: nlfilter=2
      CHARACTER (LEN=2) :: clfilter(nlfilter)
      DATA clfilter /'![',']!'/
c
c Check number of arguments
c
CIF WIN
c ATTENTION! LES LIGNES SUIVANTES DOIVENT ETRE COMMENTEES SOUS AIX
      nar = NARGS() ! Number of command line arguments, including the command
      IF (nar.lt.1) THEN
CELSE
C ATTENTION! LES LIGNES SUIVANTE DOIVENT ETRE ACTIVEES SOUS AIX
C     nar = IARGC()
C     IF (nar.lt.0) THEN
CENDIF
        CALL errmss ('WRONG NUMBER OF COMMAND LINE ARGUMENTS')
        CALL usage
        STOP 0
      ENDIF
c
```

```
c Retrieve command line option, if any, and passwords (zero or more)
c
CIF WIN
c ATTENTION! LES LIGNES SUIVANTES DOIVENT ETRE COMMENTEES SOUS AIX
      DO pos = 1, nar-1
        CALL GETARG (pos, pwd, stat)
CELIF HP-UX
C ATTENTION! LES LIGNES SUIVANTES DOIVENT ETRE COMMENTEES SOUS AIX,
C OSF1, SunOS, IRIX64 MAIS DOIVENT ETRE ACTIVEES SOUS HP-UX
C     DO pos = 1, nar+1
C       CALL GETARG (pos, pwd)
C       stat = LEN_TRIM (pwd)
CELSE
C ATTENTION! LES LIGNES SUIVANTES DOIVENT ETRE ACTIVEES SOUS AIX
C     DO pos = 1, nar
C       CALL GETARG (pos, pwd)
C       stat = LEN_TRIM (pwd)
CENDIF
c
c The GETARG function returns stat=-1 in case of error. If no error,
c pwd contains the retrieved argument and stat its length in characters.
c
        IF (stat <= 0) THEN
          CALL errmss ('Wrong command line argument')
          STOP 0
        ENDIF
        CALL to_upper (pwd)
        IF (stat > 1 .and. pwd (1:2) == '-C') THEN
c
c Option -C<character>
c
          IF (stat == 2) THEN
            echo = .FALSE.
          ELSEIF (stat == 3) THEN
            first = pwd (3:3)
          ELSE
            CALL errmss ('Wrong option -C<character>')
            STOP 0
          ENDIF
        ELSE
c
c Password
c
          npa = npa + 1
          IF (npa > npassw) THEN
            CALL errmss ('Too many passwords')
            STOP 0
          ENDIF
          CALL to_upper (pwd)
          passw (npa) = pwd
          IF (npa > 1 ) then
            DO i = 1, npa - 1
              IF (passw (npa) == passw (i)) then
                CALL errmss ('The cmdline passwords must be different')
                WRITE (blabla, '("password: ", A)') passw (i)
                CALL errmss (TRIM (blabla))
                STOP 0
              ENDIF
            END DO
          ENDIF
        ENDIF
      END DO
c
c Process input text
c
    1 READ (5, '(A)', IOSTAT=ioerror) card
c
      CALL FILTER_CARD(card, clfilter, nlfilter, len(card))
c
      IF (ioerror > 0) THEN
c
c Read error encountered
c
        CALL ERRMSS ('While reading')
        WRITE (blabla, '("on line", i6)') iline+1
        CALL errmss (TRIM (blabla))
        STOP 0
      ELSEIF (ioerror < 0) THEN
c
c End-of-file or end-of-record reached
c
        GO TO 999
      ENDIF
      iline = iline + 1
c
      c7 = card (1:7)
      CALL to_upper (c7)
c
c Skip SALOME dedicated commands
      IF (c7(1:7) == csalome) GO TO 1
c
      IF ((c7(1:4) == cif) .OR. (c7(1:4) == sif)) THEN
c
c CIF or *IF
c
        IF (LEN_TRIM (card) < 5) THEN
          CALL ERRMSS ('No password after CIF or *IF')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
        IF (.NOT. inside) THEN
! Level 1
!!hb(juillet_O9)          IF (c7(1:4) == sif) THEN
            star = .TRUE.
!!hb(juillet_O9)          ENDIF
          IF (inside2) THEN
            CALL errmss ('Conditionals are incorrectly nested')
            STOP 0
          ENDIF
          nbr = 1
          CALL get_passwords (card (5:), npt (nbr), passt (1,nbr),
     &                        iline, blabla)
          inside = .TRUE.
          CALL verify_passwords (npa, passw, npt (nbr), passt (1, nbr),
     &                           active, iline, blabla)
          IF (active) THEN
            found = .TRUE.
          ENDIF
          CALL out (echo, card, first, star, msg, active)
        ELSEIF (.NOT. inside2) THEN
! Level 2
!!hb(juillet_O9)          IF (c7(1:4) == sif) THEN
            star2 = .TRUE.
!!hb(juillet_O9)          ENDIF
          nbr2 = 1
          CALL get_passwords (card (5:), npt2 (nbr2), passt2 (1,nbr2),
     &                        iline, blabla)
          inside2 = .TRUE.
          CALL verify_passwords (npa, passw, npt2(nbr2), passt2(1,nbr2),
     &                           active2, iline, blabla)
          IF (active2) THEN
            found2 = .TRUE.
          ENDIF
          CALL out (echo, card, first, star2, msg, active.AND.active2)
        ELSE
          CALL errmss ('Nested(2) conditionals are not allowed')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
c
      ELSEIF (c7(1:7) == cifnot) THEN
c
c CIFNOT
c
        IF (LEN_TRIM (card) < 8) THEN
          CALL ERRMSS ('No password after CIFNOT')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
        IF (.NOT. inside) THEN
! Level 1
          IF (inside2) THEN
            CALL errmss ('Conditionals are incorrectly nested')
            STOP 0
          ENDIF
          nbr = 1
          CALL get_passwords (card (8:), npt (nbr), passt (1,nbr),
     &                        iline, blabla)
          inside = .TRUE.
          negative = .TRUE.
          CALL verify_passwords (npa, passw, npt (nbr), passt (1, nbr),
     &                           active, iline, blabla)
          active = .NOT. active
          IF (active) THEN
            found = .TRUE.
          ENDIF
          CALL out (echo, card, first, star, msg, active)
        ELSEIF (.NOT. inside2) THEN
! Level 2
          nbr2 = 1
          CALL get_passwords (card (8:), npt2 (nbr2), passt2 (1,nbr2),
     &                        iline, blabla)
          inside2 = .TRUE.
          negative2 = .TRUE.
          CALL verify_passwords (npa, passw, npt2(nbr2), passt2(1,nbr2),
     &                           active2, iline, blabla)
          active2 = .NOT. active2
          IF (active2) THEN
            found2 = .TRUE.
          ENDIF
          CALL out (echo, card, first, star2, msg, active.AND.active2)
        ELSE
          CALL errmss ('Nested(2) conditionals are not allowed')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
c
      ELSEIF (c7(1:6) == celif) THEN
c
c CELIF
c
        IF (LEN_TRIM (card) < 7) THEN
          CALL ERRMSS ('No password after CELIF ')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
        IF (.NOT. inside) THEN
          CALL errmss ('CELIF found while not in conditional')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
        IF (negative .OR. (inside2 .AND. negative2)) THEN
          CALL errmss ('CELIF found while inside CIFNOT')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
        IF (.NOT. inside2) THEN
! Level 1
          nbr = nbr + 1
          IF (nbr > nbran) THEN
            CALL ERRMSS ('Too many branches')
            WRITE (blabla, '("on line", i6)') iline
            CALL errmss (TRIM (blabla))
            STOP 0
          ENDIF
          CALL get_passwords (card (7:), npt (nbr), passt (1,nbr),
     &                        iline, blabla)
          CALL compare_passwords (nbr, npt, passt, iline, blabla)
          CALL verify_passwords (npa, passw, npt (nbr), passt (1, nbr),
     &                           active, iline, blabla)
          IF (active) THEN
            found = .TRUE.
          ENDIF
          CALL out (echo, card, first, star, msg, active)
        ELSE
! Level 2
          nbr2 = nbr2 + 1
          IF (nbr2 > nbran) THEN
            CALL ERRMSS ('Too many branches')
```

```
            WRITE (blabla, '("on line", i6)') iline
            CALL errmss (TRIM (blabla))
            STOP 0
          ENDIF
          CALL get_passwords (card (7:), npt2(nbr2), passt2(1,nbr2),
     &                        iline, blabla)
          CALL compare_passwords (nbr2, npt2, passt2, iline, blabla)
          CALL verify_passwords (npa, passw, npt2(nbr2), passt2(1,nbr2),
     &                           active2, iline, blabla)
          IF (active2) THEN
            found2 = .TRUE.
          ENDIF
          CALL out (echo, card, first, star2, msg, active.AND.active2)
        ENDIF
c
      ELSEIF (c7(1:6) == celse) THEN
c
c CELSE
c
        IF (inside2) THEN
! Level 2
          IF (elsed2) THEN
            CALL ERRMSS ('More than one CELSE found')
            WRITE (blabla, '("on line", i6)') iline
            CALL errmss (TRIM (blabla))
            STOP 0
          ELSE
            elsed2 = .TRUE.
          ENDIF
        ELSEIF (inside) THEN
! Level 1
          IF (elsed) THEN
            CALL ERRMSS ('More than one CELSE found')
            WRITE (blabla, '("on line", i6)') iline
            CALL errmss (TRIM (blabla))
            STOP 0
          ELSE
            elsed = .TRUE.
          ENDIF
        ENDIF
        !IF (LEN_TRIM (card) > 5) THEN
        !  CALL ERRMSS ('Extra characters after CELSE')
        !  WRITE (blabla, '("on line", i6)') iline
        !  CALL errmss (TRIM (blabla))
        !  STOP 0
        !ENDIF
        IF (.NOT. inside) THEN
          CALL errmss ('CELSE found while not in conditional')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
        IF (inside2) THEN
! Level 2
          IF (.NOT. found2) THEN
            active2 = .TRUE.
            found2 = .TRUE.
          ELSE
            active2 = .FALSE.
          ENDIF
          CALL out (echo, card, first, star2, msg, active.AND.active2)
        ELSE
! Level 1
          IF (.NOT. found) THEN
            active = .TRUE.
            found = .TRUE.
          ELSE
            active = .FALSE.
          ENDIF
          CALL out (echo, card, first, star, msg, active)
        ENDIF
c
      ELSEIF (c7(1:7) == cendif) THEN
c
c CENDIF
c
        !IF (LEN_TRIM (card) > 6) THEN
        !  CALL ERRMSS ('Extra characters after CENDIF')
        !  WRITE (blabla, '("on line", i6)') iline
        !  CALL errmss (TRIM (blabla))
        !  STOP 0
        !ENDIF
        IF (inside2) THEN
! Level 2
          inside2 = .FALSE.
          nbr2 = 0
          CALL out (echo, card, first, star2, msg, active.AND..TRUE.)
          active2 = .FALSE.
          found2 = .FALSE.
          star2 = .FALSE.
          negative2 = .FALSE.
          elsed2 = .FALSE.
        ELSEIF (inside) THEN
! Level 1
          inside = .FALSE.
          nbr = 0
          CALL out (echo, card, first, star, msg, .TRUE.)
          active = .FALSE.
          found = .FALSE.
          star = .FALSE.
          negative = .FALSE.
          elsed = .FALSE.
        ELSE
          CALL errmss ('CENDIF found while not in conditional')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
c
      ELSE
c
c Text line
c
        IF (inside2) THEN
          IF (active .AND. active2) THEN
            WRITE (6, '(A)') TRIM (card)
          ELSEIF (star2 .and. echo) THEN
            WRITE (6, '(A)') TRIM (first//card(2:))
          ENDIF
        ELSEIF (inside) THEN
          IF (active) THEN
            WRITE (6, '(A)') TRIM (card)
          ELSEIF (star .and. echo) THEN
            WRITE (6, '(A)') TRIM (first//card(2:))
          ENDIF
        ELSE
          WRITE (6, '(A)') TRIM (card)
        ENDIF
c
      ENDIF
c
      GO TO 1
c
c End of input file
c
  999 IF (inside) THEN
        CALL errmss ('EOF reached while within a CIF construct')
        STOP 0
      ENDIF
c
c Return 1 on normal execution (for use in PERL scripts).
c (When an error message is issued, the return code is 0).
c
      STOP 1
c
      END PROGRAM epx_filter
c=======================================================================
      SUBROUTINE errmss (msg)
c
      IMPLICIT NONE
c
      CHARACTER (LEN=*) :: msg
c
      WRITE (6,*) "*** ERROR *** "//msg
c
      END SUBROUTINE errmss
c=======================================================================
      SUBROUTINE usage
c
      IMPLICIT NONE
c
      WRITE (6,*) "Usage: epx_filter [-c<character>] mot1 [mot2 ...]"
      WRITE (6,*) "                  <input >output"
c
      END SUBROUTINE usage
c=======================================================================
      SUBROUTINE to_upper (s)
c
      IMPLICIT NONE
c
      CHARACTER (LEN=*), INTENT(INOUT) :: s
c
      INTEGER(4) :: i, k
c
      DO i = 1, LEN_TRIM (s)
        k = ICHAR (s (i:i))
        IF (k >= 97 .and. k <= 122) THEN
          s (i:i) = CHAR (k - 32)
        ENDIF
      END DO
c
      END SUBROUTINE to_upper
c=======================================================================
      SUBROUTINE get_passwords (card, npt, passt, iline, blabla)
c
      IMPLICIT NONE
c
      INTEGER(4), PARAMETER :: pwdlen = 16 ! Max. length of a password
      INTEGER(4), PARAMETER :: npasst = 9  ! Max. number of passwords
c                                          ! in a line of text file
c
      INTEGER(4), PARAMETER :: crdlen = 132! Max. length of a text 'card'
      CHARACTER (LEN=crdlen) :: c
c
      CHARACTER (LEN=*), INTENT(IN) :: card
      INTEGER(4), INTENT(OUT) :: npt
      CHARACTER (LEN=pwdlen),DIMENSION(npasst), INTENT(OUT) :: passt
      INTEGER(4), INTENT(IN) :: iline
      CHARACTER (LEN=*), INTENT(OUT) :: blabla
c
      INTEGER(4) :: i, k
c
      c = TRIM (card)
      npt = 0
c
      DO
        i = SCAN (TRIM (c), ' ')
        IF (i > 0) THEN
          npt = npt + 1
          IF (npt > npasst) THEN
            CALL errmss ('Too many passwords in text file')
            WRITE (blabla, '("on line", i6)') iline
            CALL errmss (TRIM (blabla))
            STOP 0
          ENDIF
          passt (npt) = c (1 : i-1)
          CALL to_upper (passt (npt))
          c = ADJUSTL (c (i+1 :))
        ELSE
          IF (LEN_TRIM (c) > 0) THEN
            npt = npt + 1
            IF (npt > npasst) THEN
              CALL errmss ('Too many passwords in text file')
              WRITE (blabla, '("on line", i6)') iline
              CALL errmss (TRIM (blabla))
              STOP 0
            ENDIF
            passt (npt) = TRIM (c)
            CALL to_upper (passt (npt))
          ENDIF
          EXIT
        ENDIF
      END DO
c
      END SUBROUTINE get_passwords
c=======================================================================
```

```
      SUBROUTINE compare_passwords (nbr, npt, passt, iline, blabla)
c
      IMPLICIT NONE
c
      INTEGER(4), PARAMETER :: pwdlen = 16 ! Max. length of a password
      INTEGER(4), PARAMETER :: npasst = 9  ! Max. number of passwords
c                                          ! in a line of text file
      INTEGER(4), PARAMETER :: nbran  = 10 ! Max. n. of branches in a CIF
c                                          ! construct
c
      INTEGER(4), INTENT(IN) :: nbr
      INTEGER(4), DIMENSION(nbran), INTENT(IN) :: npt
      CHARACTER (LEN=pwdlen),
     &          DIMENSION(npasst, nbran) :: passt  ! Passwords in each
c                                                  branch of txtfile
      INTEGER(4), INTENT(IN) :: iline
      CHARACTER (LEN=*), INTENT(OUT) :: blabla
c
      CHARACTER (LEN=pwdlen) :: passj, passl
c
      INTEGER(4) :: i, j, k, l
c
      DO i = 2, nbr
        DO j = 1, npt (i)
          passj = passt (j, i)
          DO k = 1, nbr - 1
            DO l = 1, npt (k)
              passl = passt (l, k)
              IF (l /= j .OR. k /= i) THEN
                IF (passj == passl) THEN
                  CALL errmss ('Repeated password ')
                  WRITE (blabla, '(a, "on line", I6)') passj, iline
                  CALL errmss (TRIM (blabla))
                  STOP 0
                ENDIF
              ENDIF
            END DO
          END DO
        END DO
      END DO
c
      END SUBROUTINE compare_passwords
c=======================================================================
      SUBROUTINE out (echo, card, first, star, msg, active)
c
      IMPLICIT NONE
c
      INTEGER(4), PARAMETER :: crdlen = 132! Max. length of a text 'card'
c
      LOGICAL(4), INTENT(IN) :: echo
      CHARACTER (LEN=crdlen), INTENT(IN) :: card
      CHARACTER (LEN=1), INTENT(IN) :: first
      LOGICAL(4), INTENT(IN) :: star
      CHARACTER (LEN=21), INTENT(IN) :: msg
      LOGICAL(4), INTENT(IN) :: active
c
      IF (star) THEN
        IF (echo) WRITE (6, '(A)') TRIM (first//card(2:))
      ELSE
        IF (echo) THEN
          WRITE (6, '(A)') TRIM (first//card(2:))
          IF (.NOT. active) THEN
            WRITE (6, '(A)') first//msg
          ENDIF
        ENDIF
      ENDIF
c
      END SUBROUTINE out
c=======================================================================
      SUBROUTINE verify_passwords (npa, passw, npt, passt, active,
     &                             iline, blabla)
c
      IMPLICIT NONE
c
      INTEGER(4), PARAMETER :: pwdlen = 16 ! Max. length of a password
      INTEGER(4), PARAMETER :: npassw = 9  ! Max. number of passwords
c                                          ! given in command line
      INTEGER(4), PARAMETER :: npasst = 9  ! Max. number of passwords
c                                          ! in each line of text file
c
      INTEGER(4), INTENT(IN) :: npa
      CHARACTER (LEN=pwdlen), DIMENSION(npassw), INTENT(IN) :: passw
      INTEGER(4), INTENT(IN) :: npt
      CHARACTER (LEN=pwdlen), DIMENSION(npasst), INTENT(IN) :: passt
      LOGICAL(4), INTENT(INOUT) :: active
      INTEGER(4), INTENT(IN) :: iline
      CHARACTER (LEN=80), INTENT(OUT) :: blabla
c
      INTEGER(4) :: i, j
      LOGICAL(4) :: match
      CHARACTER (LEN=pwdlen) :: passi, passj
c
c Compare current text line passwords with cmdline passwords,
c to see if there is at least one match.
c
      match = .FALSE.
      DO i = 1, npt
        passi = passt (i)
        DO j = 1, npa
          passj = passw (j)
          IF (passj == passi) THEN
            match = .TRUE.
            GO TO 1
          ENDIF
        END DO
      END DO

c
    1 IF (.NOT. active) THEN
c
c If at least one match was found, set active = .TRUE.
c
        IF (match) THEN
          active = .TRUE.
        ENDIF
      ELSE
c
c If a match was found, give error message,
c otherwise set active = .FALSE.
c
        IF (match) THEN
          CALL errmss ('Specified passwords match '//
     &                 'more than one branch')
          WRITE (blabla, '("on line ",I6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ELSE
          active = .FALSE.
        ENDIF
      ENDIF
c
      END SUBROUTINE verify_passwords
c=======================================================================
      SUBROUTINE FILTER_CARD(card, clfilter, nlfilter, ll)
*
      IMPLICIT NONE
*
      INTEGER, INTENT(IN) :: ll, nlfilter
      CHARACTER (LEN=ll), INTENT(INOUT) :: card
      CHARACTER (LEN=*), INTENT(IN) :: clfilter(nlfilter)

      CHARACTER (LEN=ll) :: card2
      INTEGER :: III, II, I, J, LJ

      card2(1:ll)=card(1:ll)
      card(1:ll)=' '
      II=0
      III=0
      LOOP_IN_CARD : DO I=1,LEN_TRIM(card2)
         IF (II > ll) EXIT
         DO J=1,nlfilter
            LJ=len(clfilter(J))
            IF (card2(II+1:II+LJ) == clfilter(J)(1:LJ)) THEN
               II=II+LJ
               CYCLE LOOP_IN_CARD
            ENDIF
         ENDDO
         II=II+1
         III=III+1
         card(III:III)=card2(II:II)
      ENDDO LOOP_IN_CARD
      card(II+1:ll)=' '

      END SUBROUTINE FILTER_CARD
```

epx_filter.ff

```
      PROGRAM epx_filter
!
! New version (fc, 7/10/2004) allowing single-level nesting of conditionals
!
! New version (fc, 7/10/2008) : Correct bug in COMPARE_PASSWORDS, now
!                               one may effectively have up to NBRAN
!                               conditional branches CELIF (as was
!                               initially foreseen), not just one!
!                               Correct bug in VERIFY_PASSWORDS : in case
!                               of error the STOP code is "0" and not "0 1"
!                               (which was returned as 1).
! New version (hb, juillet_09): Syntax A) et C) donne les memes resultats
!                               que le syntax B)
!                               le fichier filtré conserve la meme
!                               longueur de le fichier original
!                               C'est tres utile pour un debogage.
!May 2011                      : ajout du flag "%SALOME"
!July 2011                     : filtrage des chaines '![' et ']!' destinee a ai
der
!                                la mise en donnees dans SALOME
!
!
!
!
c Filters text files containing the following 3 types of construct:
c
c       Syntax A)             |    Syntax B)             |    Syntax C)
c                             |                          |
c     CIF mot1 [mot2 ...]     |  *IF mot1 [mot2 ...]     |  CIFNOT mot1 [mot2 ...]
c          [text1]            |      [text1]             |      [text1]
c    [CELIF mot3 [mot4 ...]] | [CELIF mot3 [mot4 ...]] | [CELSE]
c          [text2]            |      [text2]             |      [text2]
c    [CELSE]                  | [CELSE]                  |  CENDIF
c          [text3]            |      [text3]             |
c     CENDIF                  |  CENDIF                  |
c
c Components in [ ] are optional.
c
c The mot1, mot2 etc are passwords given on the command line by the user.
c
c The concatenation of two or more passwords corresponds to a
c logical .OR. between them. Thus CIF mot1 mot2 means IF (mot1 .OR. mot2)
c and CIFNOT mot1 mot2 means IF (.NOT. (mot1 .OR. mot2)).
c
c The CELSE and CENDIF passwords may be optionally followed by a
c comment on the same line. Comments must be separated by at least
c one blank from the preceding keyword. For example:
c CIF toto        <- 'toto' is a password here
c ELSE non-toto   <- 'non-toto' is a comment
c CENDIF toto     <- 'toto' is a comment
c
c Usage: epx_filter [-c<character>] [mot1 [mot2 ...]] <input >output
c
c
c
c For syntaxes A) and C):
c ======================
c
c        -c<character>   This option, if present, causes the use of
c                        the given <character> in place of C in the
c                        output transcription of keywords, and of
c                        the built-in message *** ACCES INTERDIT ***,
c                        that is automatically output in place
c                        of the non-activated text blocks.
c                        Example: '-c%' would give in output %IF,
c                        %ELIF, %ELSE, %ENDIF and %** ACCES INTERDIT ***.
c                        This example may be used for LaTeX documents, that
```

```
c                       use % as comment character.
c
c       -c              If no character (i.e. a blank) is specified, then
c                       neither the keyword lines nor the built-in
c                       message are transcribed, so the output file
c                       will contain just the activated text.
c
c For syntax B):
c =============
c
c The non-active branches, if any, are transcribed preceded by the chosen
c (or default) comment character. They remain therefore visible (but inactive)
c in the filtered text.
c
c       -c<character>   This option, if present, causes the use of
c                       the given <character> in place of C in the
c                       output transcription of keywords, and of
c                       (commented) inactive branches.
c
c       -c              NO EFFECT with this syntax
c
!!Stop HB(juillet_09):    Cette partie est obsolete

c Note that:
c =========
c
c    - Keywords CIF, CELIF etc. MUST start in column 1.
c
c    - Keywords are NOT case sensitive, i.e. CIF, cif or CiF are the same.
c
c    - Passwords mot1, mot2 etc. are NOT case sensitive, and may not
c      contain blanks.
c
c    - To avoid ambiguous cases, the filter performs the following tests
c      while filtering the text file:
c
c        Tests on the command line syntax:
c
c          1. The user may not specify twice the same password (or equivalent
c             passwords that differ only by letter case) on the command line.
c
c        Tests on the input text file contents:
c
c          2. Passwords in each branch of a conditional must differ from
c             any passwords in other branches of the same conditional.
c          3. If a user specifies more than one password, all passwords must
c             belong to the same branch of a conditional.
c          4. The CIFNOT syntax does not admit CELIF branches.
c
c    - The program returns 1 if everything was OK, else it returns 0
c
*IF WIN
c ATTENTION! LA LIGNE SUIVANTE DOIT ETRE COMMENTEE SOUS AIX
      USE dflib ! For GETARG, NARGS
CENDIF
c
      IMPLICIT NONE
c
      INTEGER(4), PARAMETER :: pwdlen = 16 ! Max. length of a password
      INTEGER(4), PARAMETER :: npassw = 9  ! Max. number of passwords
c                                          ! given in command line
      INTEGER(4), PARAMETER :: crdlen = 132! Max. length of a text 'card'
c
      INTEGER(4) :: nar, npa = 0, iline = 0
      INTEGER(4) :: i
*IF WIN
c ATTENTION! LA LIGNE SUIVANTE DOIT ETRE COMMENTEE SOUS AIX
      INTEGER(2) :: pos, stat
CELSE
c ATTENTION! LA LIGNE SUIVANTE DOIT ETRE ACTIVEE SOUS AIX
      INTEGER(4) :: pos, stat, iargc
CENDIF
      CHARACTER (LEN=pwdlen) :: pwd
      CHARACTER (LEN=pwdlen), DIMENSION(npassw) :: passw  ! Passwords in cmdline
      CHARACTER (LEN=1) :: first = 'C'
      LOGICAL(4) :: echo = .TRUE., inside = .FALSE.
      LOGICAL(4) :: active = .FALSE., found = .FALSE.
      LOGICAL(4) :: star = .FALSE., negative = .FALSE.
      LOGICAL(4) :: elsed = .FALSE.
!
      LOGICAL(4) :: inside2 = .FALSE.
      LOGICAL(4) :: active2 = .FALSE., found2 = .FALSE.
      LOGICAL(4) :: star2 = .FALSE., negative2 = .FALSE.
      LOGICAL(4) :: elsed2 = .FALSE.
c
c echo   : TRUE  if keywords and built-in message are to be transcribed,
c                preceded by the comment character
c          FALSE otherwise
c
c inside : TRUE  if we are inside a CIF or *IF or CIFNOT construct
c          FALSE otherwise
c
c active : TRUE  if we are in the active branch of the construct
c          FALSE otherwise
c
c star   : TRUE  if we are in an *IF construct
c          FALSE otherwise
c
c negative: TRUE  if we are in a CIFNOT construct
c           FALSE otherwise
!
! xxx2 : same meaning as xxx but for the nested conditional
!
! We assume throughout that inside2 implies inside !!!!!!!!!!!!!!!!!!!!
c
      CHARACTER (LEN=crdlen) :: card
      CHARACTER (LEN=4) :: cif    = "CIF "
      CHARACTER (LEN=4) :: sif    = "*IF "
      CHARACTER (LEN=7) :: cifnot = "CIFNOT "
      CHARACTER (LEN=6) :: celif  = "CELIF "
      CHARACTER (LEN=6) :: celse  = "CELSE "
      CHARACTER (LEN=7) :: cendif = "CENDIF "
      CHARACTER (LEN=7) :: c7
      CHARACTER (LEN=7) :: csalome= "%SALOME"
c
      INTEGER(4), PARAMETER :: npasst = 9  ! Max. number of passwords
c                                          ! in each line of text file
      INTEGER(4), PARAMETER :: nbran  = 10 ! Max. n. of branches in a CIF
c                                          ! construct
      INTEGER(4), DIMENSION(nbran) :: npt = 0
      INTEGER(4) :: nbr = 0
      CHARACTER (LEN=pwdlen),
     &          DIMENSION(npasst, nbran) :: passt  ! Passwords in each
c                                                  branch of txtfile
! same as above but for the nested conditional
      INTEGER(4), DIMENSION(nbran) :: npt2 = 0
      INTEGER(4) :: nbr2 = 0
      CHARACTER (LEN=pwdlen),
     &          DIMENSION(npasst, nbran) :: passt2 ! Passwords in each
c                                                  branch of txtfile
!
      CHARACTER (LEN=21) :: msg = "** ACCES INTERDIT ***"
c
      CHARACTER (LEN=80) :: blabla
      INTEGER (4) :: ioerror
c
      INTEGER, PARAMETER :: nlfilter=2
      CHARACTER (LEN=2) :: clfilter(nlfilter)
      DATA clfilter /'![',']!'/
c
c Check number of arguments
c
*IF WIN
c ATTENTION! LES LIGNES SUIVANTES DOIVENT ETRE COMMENTEES SOUS AIX
      nar = NARGS() ! Number of command line arguments, including the command
      IF (nar.lt.1) THEN
CELSE
c ATTENTION! LES LIGNES SUIVANTE DOIVENT ETRE ACTIVEES SOUS AIX
      nar = IARGC()
      IF (nar.lt.0) THEN
CENDIF
        CALL errmss ('WRONG NUMBER OF COMMAND LINE ARGUMENTS')
        CALL usage
        STOP 0
      ENDIF
c
c Retrieve command line option, if any, and passwords (zero or more)
c
*IF WIN
c ATTENTION! LES LIGNES SUIVANTES DOIVENT ETRE COMMENTEES SOUS AIX
      DO pos = 1, nar-1
        CALL GETARG (pos, pwd, stat)
CELIF HP-UX
c ATTENTION! LES LIGNES SUIVANTES DOIVENT ETRE COMMENTEES SOUS AIX,
c OSF1, SunOS, IRIX64 MAIS DOIVENT ETRE ACTIVEES SOUS HP-UX
      DO pos = 1, nar+1
        CALL GETARG (pos, pwd)
        stat = LEN_TRIM (pwd)
CELSE
c ATTENTION! LES LIGNES SUIVANTES DOIVENT ETRE ACTIVEES SOUS AIX
      DO pos = 1, nar
        CALL GETARG (pos, pwd)
        stat = LEN_TRIM (pwd)
CENDIF
c
c The GETARG function returns stat=-1 in case of error. If no error,
c pwd contains the retrieved argument and stat its length in characters.
c
        IF (stat <= 0) THEN
          CALL errmss ('Wrong command line argument')
          STOP 0
        ENDIF
        CALL to_upper (pwd)
        IF (stat > 1 .and. pwd (1:2) == '-C') THEN
c
c Option -C<character>
c
          IF (stat == 2) THEN
            echo = .FALSE.
          ELSEIF (stat == 3) THEN
            first = pwd (3:3)
          ELSE
            CALL errmss ('Wrong option -C<character>')
            STOP 0
          ENDIF
        ELSE
c
c Password
c
          npa = npa + 1
          IF (npa > npassw) THEN
            CALL errmss ('Too many passwords')
            STOP 0
          ENDIF
          CALL to_upper (pwd)
          passw (npa) = pwd
          IF (npa > 1 ) then
            DO i = 1, npa - 1
              IF (passw (npa) == passw (i)) then
                CALL errmss ('The cmdline passwords must be different')
                WRITE (blabla, '("password: ", A)') passw (i)
                CALL errmss (TRIM (blabla))
                STOP 0
              ENDIF
            END DO
          ENDIF
        ENDIF
      END DO
c
c Process input text
c
    1 READ (5, '(A)', IOSTAT=ioerror) card
c
      CALL FILTER_CARD(card, clfilter, nlfilter, len(card))
c
      IF (ioerror > 0) THEN
c
c Read error encountered
c
        CALL ERRMSS ('While reading')
        WRITE (blabla, '("on line", i6)') iline+1
        CALL errmss (TRIM (blabla))
        STOP 0
      ELSEIF (ioerror < 0) THEN
c
c End-of-file or end-of-record reached
```

```
c
        GO TO 999
      ENDIF
      iline = iline + 1
c
      c7 = card (1:7)
      CALL to_upper (c7)
c
c Skip SALOME dedicated commands
      IF (c7(1:7) == csalome) GO TO 1
c
      IF ((c7(1:4) == cif) .OR. (c7(1:4) == sif)) THEN
c
c CIF or *IF
c
        IF (LEN_TRIM (card) < 5) THEN
          CALL ERRMSS ('No password after CIF or *IF')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
        IF (.NOT. inside) THEN
! Level 1
!!hb(juillet_O9)          IF (c7(1:4) == sif) THEN
            star = .TRUE.
!!hb(juillet_O9)          ENDIF
          IF (inside2) THEN
            CALL errmss ('Conditionals are incorrectly nested')
            STOP 0
          ENDIF
          nbr = 1
          CALL get_passwords (card (5:), npt (nbr), passt (1,nbr),
     &                        iline, blabla)
          inside = .TRUE.
          CALL verify_passwords (npa, passw, npt (nbr), passt (1, nbr),
     &                           active, iline, blabla)
          IF (active) THEN
            found = .TRUE.
          ENDIF
          CALL out (echo, card, first, star, msg, active)
        ELSEIF (.NOT. inside2) THEN
! Level 2
!!hb(juillet_O9)          IF (c7(1:4) == sif) THEN
            star2 = .TRUE.
!!hb(juillet_O9)          ENDIF
          nbr2 = 1
          CALL get_passwords (card (5:), npt2 (nbr2), passt2 (1,nbr2),
     &                        iline, blabla)
          inside2 = .TRUE.
          CALL verify_passwords (npa, passw, npt2(nbr2), passt2(1,nbr2),
     &                           active2, iline, blabla)
          IF (active2) THEN
            found2 = .TRUE.
          ENDIF
          CALL out (echo, card, first, star2, msg, active.AND.active2)
        ELSE
          CALL errmss ('Nested(2) conditionals are not allowed')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
c
      ELSEIF (c7(1:7) == cifnot) THEN
c
c CIFNOT
c
        IF (LEN_TRIM (card) < 8) THEN
          CALL ERRMSS ('No password after CIFNOT')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
        IF (.NOT. inside) THEN
! Level 1
          IF (inside2) THEN
            CALL errmss ('Conditionals are incorrectly nested')
            STOP 0
          ENDIF
          nbr = 1
          CALL get_passwords (card (8:), npt (nbr), passt (1,nbr),
     &                        iline, blabla)
          inside = .TRUE.
          negative = .TRUE.
          CALL verify_passwords (npa, passw, npt (nbr), passt (1, nbr),
     &                           active, iline, blabla)
          active = .NOT. active
          IF (active) THEN
            found = .TRUE.
          ENDIF
          CALL out (echo, card, first, star, msg, active)
        ELSEIF (.NOT. inside2) THEN
! Level 2
          nbr2 = 1
          CALL get_passwords (card (8:), npt2 (nbr2), passt2 (1,nbr2),
     &                        iline, blabla)
          inside2 = .TRUE.
          negative2 = .TRUE.
          CALL verify_passwords (npa, passw, npt2(nbr2), passt2(1,nbr2),
     &                           active2, iline, blabla)
          active2 = .NOT. active2
          IF (active2) THEN
            found2 = .TRUE.
          ENDIF
          CALL out (echo, card, first, star2, msg, active.AND.active2)
        ELSE
          CALL errmss ('Nested(2) conditionals are not allowed')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
c
      ELSEIF (c7(1:6) == celif) THEN
c
c CELIF
c
        IF (LEN_TRIM (card) < 7) THEN
          CALL ERRMSS ('No password after CELIF ')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
        IF (.NOT. inside) THEN
          CALL errmss ('CELIF found while not in conditional')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
        IF (negative .OR. (inside2 .AND. negative2)) THEN
          CALL errmss ('CELIF found while inside CIFNOT')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
        IF (.NOT. inside2) THEN
! Level 1
          nbr = nbr + 1
          IF (nbr > nbran) THEN
            CALL ERRMSS ('Too many branches')
            WRITE (blabla, '("on line", i6)') iline
            CALL errmss (TRIM (blabla))
            STOP 0
          ENDIF
          CALL get_passwords (card (7:), npt (nbr), passt (1,nbr),
     &                        iline, blabla)
          CALL compare_passwords (nbr, npt, passt, iline, blabla)
          CALL verify_passwords (npa, passw, npt (nbr), passt (1, nbr),
     &                           active, iline, blabla)
          IF (active) THEN
            found = .TRUE.
          ENDIF
          CALL out (echo, card, first, star, msg, active)
        ELSE
! Level 2
          nbr2 = nbr2 + 1
          IF (nbr2 > nbran) THEN
            CALL ERRMSS ('Too many branches')
            WRITE (blabla, '("on line", i6)') iline
            CALL errmss (TRIM (blabla))
            STOP 0
          ENDIF
          CALL get_passwords (card (7:), npt2(nbr2), passt2(1,nbr2),
     &                        iline, blabla)
          CALL compare_passwords (nbr2, npt2, passt2, iline, blabla)
          CALL verify_passwords (npa, passw, npt2(nbr2), passt2(1,nbr2),
     &                           active2, iline, blabla)
          IF (active2) THEN
            found2 = .TRUE.
          ENDIF
          CALL out (echo, card, first, star2, msg, active.AND.active2)
        ENDIF
c
      ELSEIF (c7(1:6) == celse) THEN
c
c CELSE
c
        IF (inside2) THEN
! Level 2
          IF (elsed2) THEN
            CALL ERRMSS ('More than one CELSE found')
            WRITE (blabla, '("on line", i6)') iline
            CALL errmss (TRIM (blabla))
            STOP 0
          ELSE
            elsed2 = .TRUE.
          ENDIF
        ELSEIF (inside) THEN
! Level 1
          IF (elsed) THEN
            CALL ERRMSS ('More than one CELSE found')
            WRITE (blabla, '("on line", i6)') iline
            CALL errmss (TRIM (blabla))
            STOP 0
          ELSE
            elsed = .TRUE.
          ENDIF
        ENDIF
        !IF (LEN_TRIM (card) > 5) THEN
        !  CALL ERRMSS ('Extra characters after CELSE')
        !  WRITE (blabla, '("on line", i6)') iline
        !  CALL errmss (TRIM (blabla))
        !  STOP 0
        !ENDIF
        IF (.NOT. inside) THEN
          CALL errmss ('CELSE found while not in conditional')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
        IF (inside2) THEN
! Level 2
          IF (.NOT. found2) THEN
            active2 = .TRUE.
            found2 = .TRUE.
          ELSE
            active2 = .FALSE.
          ENDIF
          CALL out (echo, card, first, star2, msg, active.AND.active2)
        ELSE
! Level 1
          IF (.NOT. found) THEN
            active = .TRUE.
            found = .TRUE.
          ELSE
            active = .FALSE.
          ENDIF
          CALL out (echo, card, first, star, msg, active)
        ENDIF
c
      ELSEIF (c7(1:7) == cendif) THEN
c
c CENDIF
c
        !IF (LEN_TRIM (card) > 6) THEN
        !  CALL ERRMSS ('Extra characters after CENDIF')
        !  WRITE (blabla, '("on line", i6)') iline
        !  CALL errmss (TRIM (blabla))
```

```
        !  STOP 0
        !ENDIF
        IF (inside2) THEN
! Level 2
          inside2 = .FALSE.
          nbr2 = 0
          CALL out (echo, card, first, star2, msg, active.AND..TRUE.)
          active2 = .FALSE.
          found2 = .FALSE.
          star2 = .FALSE.
          negative2 = .FALSE.
          elsed2 = .FALSE.
        ELSEIF (inside) THEN
! Level 1
          inside = .FALSE.
          nbr = 0
          CALL out (echo, card, first, star, msg, .TRUE.)
          active = .FALSE.
          found = .FALSE.
          star = .FALSE.
          negative = .FALSE.
          elsed = .FALSE.
        ELSE
          CALL errmss ('CENDIF found while not in conditional')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
c
      ELSE
c
c Text line
c
        IF (inside2) THEN
          IF (active .AND. active2) THEN
            WRITE (6, '(A)') TRIM (card)
          ELSEIF (star2 .and. echo) THEN
            WRITE (6, '(A)') TRIM (first//card(2:))
          ENDIF
        ELSEIF (inside) THEN
          IF (active) THEN
            WRITE (6, '(A)') TRIM (card)
          ELSEIF (star .and. echo) THEN
            WRITE (6, '(A)') TRIM (first//card(2:))
          ENDIF
        ELSE
          WRITE (6, '(A)') TRIM (card)
        ENDIF
c
      ENDIF
c
      GO TO 1
c
c End of input file
c
  999 IF (inside) THEN
        CALL errmss ('EOF reached while within a CIF construct')
        STOP 0
      ENDIF
c
c Return 1 on normal execution (for use in PERL scripts).
c (When an error message is issued, the return code is 0).
c
      STOP 1
c
      END PROGRAM epx_filter
c=======================================================================
      SUBROUTINE errmss (msg)
c
      IMPLICIT NONE
c
      CHARACTER (LEN=*) :: msg
c
      WRITE (6,*) "*** ERROR *** "//msg
c
      END SUBROUTINE errmss
c=======================================================================
      SUBROUTINE usage
c
      IMPLICIT NONE
c
      WRITE (6,*) "Usage: epx_filter [-c<character>] mot1 [mot2 ...]"
      WRITE (6,*) "                  <input >output"
c
      END SUBROUTINE usage
c=======================================================================
      SUBROUTINE to_upper (s)
c
      IMPLICIT NONE
c
      CHARACTER (LEN=*), INTENT(INOUT) :: s
c
      INTEGER(4) :: i, k
c
      DO i = 1, LEN_TRIM (s)
        k = ICHAR (s (i:i))
        IF (k >= 97 .and. k <= 122) THEN
          s (i:i) = CHAR (k - 32)
        ENDIF
      END DO
c
      END SUBROUTINE to_upper
c=======================================================================
      SUBROUTINE get_passwords (card, npt, passt, iline, blabla)
c
      IMPLICIT NONE
c
      INTEGER(4), PARAMETER :: pwdlen = 16 ! Max. length of a password
      INTEGER(4), PARAMETER :: npasst = 9  ! Max. number of passwords
c                                          ! in a line of text file
c
      INTEGER(4), PARAMETER :: crdlen = 132! Max. length of a text 'card'
      CHARACTER (LEN=crdlen) :: c
c
      CHARACTER (LEN=*), INTENT(IN) :: card
      INTEGER(4), INTENT(OUT) :: npt
      CHARACTER (LEN=pwdlen),DIMENSION(npasst), INTENT(OUT) :: passt
      INTEGER(4), INTENT(IN) :: iline
      CHARACTER (LEN=*), INTENT(OUT) :: blabla
c
      INTEGER(4) :: i, k
c
      c = TRIM (card)
      npt = 0
c
      DO
        i = SCAN (TRIM (c), ' ')
        IF (i > 0) THEN
          npt = npt + 1
          IF (npt > npasst) THEN
            CALL errmss ('Too many passwords in text file')
            WRITE (blabla, '("on line", i6)') iline
            CALL errmss (TRIM (blabla))
            STOP 0
          ENDIF
          passt (npt) = c (1 : i-1)
          CALL to_upper (passt (npt))
          c = ADJUSTL (c (i+1 :))
        ELSE
          IF (LEN_TRIM (c) > 0) THEN
            npt = npt + 1
            IF (npt > npasst) THEN
              CALL errmss ('Too many passwords in text file')
              WRITE (blabla, '("on line", i6)') iline
              CALL errmss (TRIM (blabla))
              STOP 0
            ENDIF
            passt (npt) = TRIM (c)
            CALL to_upper (passt (npt))
          ENDIF
          EXIT
        ENDIF
      END DO
c
      END SUBROUTINE get_passwords
c=======================================================================
      SUBROUTINE compare_passwords (nbr, npt, passt, iline, blabla)
c
      IMPLICIT NONE
c
      INTEGER(4), PARAMETER :: pwdlen = 16 ! Max. length of a password
      INTEGER(4), PARAMETER :: npasst = 9  ! Max. number of passwords
c                                          ! in a line of text file
      INTEGER(4), PARAMETER :: nbran  = 10 ! Max. n. of branches in a CIF
c                                          ! construct
c
      INTEGER(4), INTENT(IN) :: nbr
      INTEGER(4), DIMENSION(nbran), INTENT(IN) :: npt
      CHARACTER (LEN=pwdlen),
     &           DIMENSION(npasst, nbran) :: passt  ! Passwords in each
c                                                    branch of txtfile
      INTEGER(4), INTENT(IN) :: iline
      CHARACTER (LEN=*), INTENT(OUT) :: blabla
c
      CHARACTER (LEN=pwdlen) :: passj, passl
c
      INTEGER(4) :: i, j, k, l
c
      DO i = 2, nbr
        DO j = 1, npt (i)
          passj = passt (j, i)
          DO k = 1, nbr - 1
            DO l = 1, npt (k)
              passl = passt (l, k)
              IF (l /= j .OR. k /= i) THEN
                IF (passj == passl) THEN
                  CALL errmss ('Repeated password ')
                  WRITE (blabla, '(a, "on line", I6)') passj, iline
                  CALL errmss (TRIM (blabla))
                  STOP 0
                ENDIF
              ENDIF
            END DO
          END DO
        END DO
      END DO
c
      END SUBROUTINE compare_passwords
c=======================================================================
      SUBROUTINE out (echo, card, first, star, msg, active)
c
      IMPLICIT NONE
c
      INTEGER(4), PARAMETER :: crdlen = 132! Max. length of a text 'card'
c
      LOGICAL(4), INTENT(IN) :: echo
      CHARACTER (LEN=crdlen), INTENT(IN) :: card
      CHARACTER (LEN=1), INTENT(IN) :: first
      LOGICAL(4), INTENT(IN) :: star
      CHARACTER (LEN=21), INTENT(IN) :: msg
      LOGICAL(4), INTENT(IN) :: active
c
      IF (star) THEN
        IF (echo) WRITE (6, '(A)') TRIM (first//card(2:))
      ELSE
        IF (echo) THEN
          WRITE (6, '(A)') TRIM (first//card(2:))
          IF (.NOT. active) THEN
            WRITE (6, '(A)') first//msg
          ENDIF
        ENDIF
      ENDIF
c
      END SUBROUTINE out
c=======================================================================
      SUBROUTINE verify_passwords (npa, passw, npt, passt, active,
     &                             iline, blabla)
c
      IMPLICIT NONE
c
      INTEGER(4), PARAMETER :: pwdlen = 16 ! Max. length of a password
      INTEGER(4), PARAMETER :: npassw = 9  ! Max. number of passwords
c                                          ! given in command line
      INTEGER(4), PARAMETER :: npasst = 9  ! Max. number of passwords
c                                          ! in each line of text file
c
```

```
      INTEGER(4), INTENT(IN) :: npa
      CHARACTER (LEN=pwdlen), DIMENSION(npassw), INTENT(IN) :: passw
      INTEGER(4), INTENT(IN) :: npt
      CHARACTER (LEN=pwdlen), DIMENSION(npasst), INTENT(IN) :: passt
      LOGICAL(4), INTENT(INOUT) :: active
      INTEGER(4), INTENT(IN) :: iline
      CHARACTER (LEN=80), INTENT(OUT) :: blabla
c
      INTEGER(4) :: i, j
      LOGICAL(4) :: match
      CHARACTER (LEN=pwdlen) :: passi, passj
c
c Compare current text line passwords with cmdline passwords,
c to see if there is at least one match.
c
      match = .FALSE.
      DO i = 1, npt
        passi = passt (i)
        DO j = 1, npa
          passj = passw (j)
          IF (passj == passi) THEN
            match = .TRUE.
            GO TO 1
          ENDIF
        END DO
      END DO

c
    1 IF (.NOT. active) THEN
c
c If at least one match was found, set active = .TRUE.
c
        IF (match) THEN
          active = .TRUE.
        ENDIF
      ELSE
c
c If a match was found, give error message,
c otherwise set active = .FALSE.
c
        IF (match) THEN
          CALL errmss ('Specified passwords match '//
     &                 'more than one branch')
          WRITE (blabla, '("on line ",I6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ELSE
          active = .FALSE.
        ENDIF
      ENDIF
c
      END SUBROUTINE verify_passwords
c=======================================================================
      SUBROUTINE FILTER_CARD(card, clfilter, nlfilter, ll)
*
      IMPLICIT NONE
*
      INTEGER, INTENT(IN) :: ll, nlfilter
      CHARACTER (LEN=ll), INTENT(INOUT) :: card
      CHARACTER (LEN=*), INTENT(IN) :: clfilter(nlfilter)

      CHARACTER (LEN=ll) :: card2
      INTEGER :: III, II, I, J, LJ

      card2(1:ll)=card(1:ll)
      card(1:ll)=' '
      II=0
      III=0
      LOOP_IN_CARD : DO I=1,LEN_TRIM(card2)
         IF (II > ll) EXIT
         DO J=1,nlfilter
            LJ=len(clfilter(J))
            IF (card2(II+1:II+LJ) == clfilter(J)(1:LJ)) THEN
               II=II+LJ
               CYCLE LOOP_IN_CARD
            ENDIF
         ENDDO
         II=II+1
         III=III+1
         card(III:III)=card2(II:II)
      ENDDO LOOP_IN_CARD
      card(II+1:ll)=' '

      END SUBROUTINE FILTER_CARD
```

epx_filter_aix.f

```
      PROGRAM epx_filter
!
! New version (fc, 7/10/2004) allowing single-level nesting of conditionals
!
! New version (fc, 7/10/2008) : Correct bug in COMPARE_PASSWORDS, now
!                               one may effectively have up to NBRAN
!                               conditional branches CELIF (as was
!                               initially foreseen), not just one!
!                               Correct bug in VERIFY_PASSWORDS : in case
!                               of error the STOP code is "0" and not "0 1"
!                               (which was returned as 1).

!hb Oct_08 : for non-Windows platforms:
!                    code error  OK      => STOP 'OK'
!                                else    => STOP 'FAILED'
!
c
c Filters text files containing the following 3 types of construct:
c
c       Syntax A)             |    Syntax B)             |    Syntax C)
c                             |                          |
c     CIF mot1 [mot2 ...]     |  *IF mot1 [mot2 ...]     |  CIFNOT mot1 [mot2 ...]
c         [text1]             |      [text1]             |      [text1]
c    [CELIF mot3 [mot4 ...]] | [CELIF mot3 [mot4 ...]] | [CELSE]
c         [text2]             |      [text2]             |      [text2]
c     [CELSE]                 | [CELSE]                  |  CENDIF
c         [text3]             |      [text3]             |
c     CENDIF                  |  CENDIF                  |
c
c Components in [ ] are optional.
c
c The mot1, mot2 etc are passwords given on the command line by the user.
c
c The concatenation of two or more passwords corresponds to a
c logical .OR. between them. Thus CIF mot1 mot2 means IF (mot1 .OR. mot2)
c and CIFNOT mot1 mot2 means IF (.NOT. (mot1 .OR. mot2)).
c
c The CELSE and CENDIF passwords may be optionally followed by a
c comment on the same line. Comments must be separated by at least
c one blank from the preceding keyword. For example:
c CIF toto        <- 'toto' is a password here
c ELSE non-toto   <- 'non-toto' is a comment
c CENDIF toto     <- 'toto' is a comment
c
c Usage: epx_filter [-c<character>] [mot1 [mot2 ...]] <input >output
c
c
c For syntaxes A) and C):
c =====================
c
c        -c<character>   This option, if present, causes the use of
c                        the given <character> in place of C in the
c                        output transcription of keywords, and of
c                        the built-in message *** ACCES INTERDIT ***,
c                        that is automatically output in place
c                        of the non-activated text blocks.
c                        Example: '-c%' would give in output %IF,
c                        %ELIF, %ELSE, %ENDIF and %** ACCES INTERDIT ***.
c                        This example may be used for LaTeX documents, that
c                        use % as comment character.
c
c        -c              If no character (i.e. a blank) is specified, then
c                        neither the keyword lines nor the built-in
c                        message are transcribed, so the output file
c                        will contain just the activated text.
c
c For syntax B):
c =============
c
c The non-active branches, if any, are transcribed preceded by the chosen
c (or default) comment character. They remain therefore visible (but inactive)
c in the filtered text.
c
c        -c<character>   This option, if present, causes the use of
c                        the given <character> in place of C in the
c                        output transcription of keywords, and of
c                        (commented) inactive branches.
c
c        -c              NO EFFECT with this syntax
c
c Note that:
c ========
c
c    - Keywords CIF, CELIF etc. MUST start in column 1.
c
c    - Keywords are NOT case sensitive, i.e. CIF, cif or CiF are the same.
c
c    - Passwords mot1, mot2 etc. are NOT case sensitive, and may not
c      contain blanks.
c
c    - To avoid ambiguous cases, the filter performs the following tests
c      while filtering the text file:
c
c        Tests on the command line syntax:
c
c          1. The user may not specify twice the same password (or equivalent
c             passwords that differ only by letter case) on the command line.
c
c        Tests on the input text file contents:
c
c          2. Passwords in each branch of a conditional must differ from
c             any passwords in other branches of the same conditional.
c          3. If a user specifies more than one password, all passwords must
c             belong to the same branch of a conditional.
c          4. The CIFNOT syntax does not admit CELIF branches.
c
c    - The program returns 1 if everything was OK, else it returns 0
c
*IF WIN
c ATTENTION! LA LIGNE SUIVANTE DOIT ETRE COMMENTEE SOUS AIX
!     USE dflib ! For GETARG, NARGS
CENDIF
c
      IMPLICIT NONE
c
      INTEGER(4), PARAMETER :: pwdlen = 16 ! Max. length of a password
      INTEGER(4), PARAMETER :: npassw = 8  ! Max. number of passwords
c                                          ! given in command line
      INTEGER(4), PARAMETER :: crdlen = 132! Max. length of a text 'card'
c
      INTEGER(4) :: nar, npa = 0, iline = 0
      INTEGER(4) :: i
*IF WIN
c ATTENTION! LA LIGNE SUIVANTE DOIT ETRE COMMENTEE SOUS AIX
!     INTEGER(2) :: pos, stat
CELSE
c ATTENTION! LA LIGNE SUIVANTE DOIT ETRE ACTIVEE SOUS AIX
      INTEGER(4) :: pos, stat, iargc
CENDIF
      CHARACTER (LEN=pwdlen) :: pwd
      CHARACTER (LEN=pwdlen), DIMENSION(npassw) :: passw  ! Passwords in cmdline
      CHARACTER (LEN=1) :: first = 'C'
      LOGICAL(4) :: echo = .TRUE., inside = .FALSE.
      LOGICAL(4) :: active = .FALSE., found = .FALSE.
      LOGICAL(4) :: star = .FALSE., negative = .FALSE.
      LOGICAL(4) :: elsed = .FALSE.
!
      LOGICAL(4) :: inside2 = .FALSE.
      LOGICAL(4) :: active2 = .FALSE., found2 = .FALSE.
      LOGICAL(4) :: star2 = .FALSE., negative2 = .FALSE.
      LOGICAL(4) :: elsed2 = .FALSE.
c
c echo   : TRUE  if keywords and built-in message are to be transcribed,
c                preceded by the comment character
c          FALSE otherwise
c
c inside : TRUE  if we are inside a CIF or *IF or CIFNOT construct
```

```
c          FALSE otherwise
c
c active : TRUE  if we are in the active branch of the construct
c          FALSE otherwise
c
c star   : TRUE  if we are in an *IF construct
c          FALSE otherwise
c
c negative: TRUE  if we are in a CIFNOT construct
c           FALSE otherwise
!
! xxx2 : same meaning as xxx but for the nested conditional
!
! We assume throughout that inside2 implies inside !!!!!!!!!!!!!!!!!!!!
c
      CHARACTER (LEN=crdlen) :: card
      CHARACTER (LEN=4) :: cif    = "CIF "
      CHARACTER (LEN=4) :: sif    = "*IF "
      CHARACTER (LEN=7) :: cifnot = "CIFNOT "
      CHARACTER (LEN=6) :: celif  = "CELIF "
      CHARACTER (LEN=6) :: celse  = "CELSE "
      CHARACTER (LEN=7) :: cendif = "CENDIF "
      CHARACTER (LEN=7) :: c7
c
      INTEGER(4), PARAMETER :: npasst = 8  ! Max. number of passwords
c                                          ! in each line of text file
      INTEGER(4), PARAMETER :: nbran  = 10 ! Max. n. of branches in a CIF
c                                          ! construct
      INTEGER(4), DIMENSION(nbran) :: npt = 0
      INTEGER(4) :: nbr = 0
      CHARACTER (LEN=pwdlen),
     &          DIMENSION(npasst, nbran) :: passt  ! Passwords in each
c                                                  branch of txtfile
! same as above but for the nested conditional
      INTEGER(4), DIMENSION(nbran) :: npt2 = 0
      INTEGER(4) :: nbr2 = 0
      CHARACTER (LEN=pwdlen),
     &          DIMENSION(npasst, nbran) :: passt2 ! Passwords in each
c                                                  branch of txtfile
!
      CHARACTER (LEN=21) :: msg = "** ACCES INTERDIT ***"
c
      CHARACTER (LEN=80) :: blabla
      INTEGER (4) :: ioerror
c
c Check number of arguments
c
*IF WIN
c ATTENTION! LES LIGNES SUIVANTES DOIVENT ETRE COMMENTEES SOUS AIX
!     nar = NARGS() ! Number of command line arguments, including the command
!     IF (nar.lt.1) THEN
CELSE
c ATTENTION! LES LIGNES SUIVANTE DOIVENT ETRE ACTIVEES SOUS AIX
      nar = IARGC()
      IF (nar.lt.0) THEN
CENDIF
        CALL errmss ('WRONG NUMBER OF COMMAND LINE ARGUMENTS')
        CALL usage
        CALL STOP_BY_CODE (0)
      ENDIF
c
c Retrieve command line option, if any, and passwords (zero or more)
c
*IF WIN
c ATTENTION! LES LIGNES SUIVANTES DOIVENT ETRE COMMENTEES SOUS AIX
!     DO pos = 1, nar-1
!       CALL GETARG (pos, pwd, stat)
CELIF HP-UX
c ATTENTION! LES LIGNES SUIVANTES DOIVENT ETRE COMMENTEES SOUS AIX,
c OSF1, SunOS, IRIX64 MAIS DOIVENT ETRE ACTIVEES SOUS HP-UX
!     DO pos = 1, nar+1
!       CALL GETARG (pos, pwd)
!       stat = LEN_TRIM (pwd)
CELSE
c ATTENTION! LES LIGNES SUIVANTES DOIVENT ETRE ACTIVEES SOUS AIX
      DO pos = 1, nar
        CALL GETARG (pos, pwd)
        stat = LEN_TRIM (pwd)
CENDIF
c
c The GETARG function returns stat=-1 in case of error. If no error,
c pwd contains the retrieved argument and stat its length in characters.
c
        IF (stat <= 0) THEN
          CALL errmss ('Wrong command line argument')
          CALL STOP_BY_CODE (0)
        ENDIF
        CALL to_upper (pwd)
        IF (stat > 1 .and. pwd (1:2) == '-C') THEN
c
c Option -C<character>
c
          IF (stat == 2) THEN
            echo = .FALSE.
          ELSEIF (stat == 3) THEN
            first = pwd (3:3)
          ELSE
            CALL errmss ('Wrong option -C<character>')
            CALL STOP_BY_CODE (0)
          ENDIF
        ELSE
c
c Password
c
          npa = npa + 1
          IF (npa > npassw) THEN
            CALL errmss ('Too many passwords')
            CALL STOP_BY_CODE (0)
          ENDIF
          CALL to_upper (pwd)
          passw (npa) = pwd
          IF (npa > 1 ) then
            DO i = 1, npa - 1
              IF (passw (npa) == passw (i)) then
                CALL errmss ('The cmdline passwords must be different')
                WRITE (blabla, '("password: ", A)') passw (i)
                CALL errmss (TRIM (blabla))
                CALL STOP_BY_CODE (0)
              ENDIF
            END DO
          ENDIF
        ENDIF
      END DO
c
c Process input text
c
    1 READ (5, '(A)', IOSTAT=ioerror) card
      IF (ioerror > 0) THEN
c
c Read error encountered
c
        CALL ERRMSS ('While reading')
        WRITE (blabla, '("on line", i6)') iline+1
        CALL errmss (TRIM (blabla))
        CALL STOP_BY_CODE (0)
      ELSEIF (ioerror < 0) THEN
c
c End-of-file or end-of-record reached
c
        GO TO 999
      ENDIF
      iline = iline + 1
c
      c7 = card (1:7)
      CALL to_upper (c7)
c
      IF ((c7(1:4) == cif) .OR. (c7(1:4) == sif)) THEN
c
c CIF or *IF
c
        IF (LEN_TRIM (card) < 5) THEN
          CALL ERRMSS ('No password after CIF or *IF')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          CALL STOP_BY_CODE (0)
        ENDIF
        IF (.NOT. inside) THEN
! Level 1
          IF (c7(1:4) == sif) THEN
            star = .TRUE.
          ENDIF
          IF (inside2) THEN
            CALL errmss ('Conditionals are incorrectly nested')
            CALL STOP_BY_CODE (0)
          ENDIF
          nbr = 1
          CALL get_passwords (card (5:), npt (nbr), passt (1,nbr),
     &                        iline, blabla)
          inside = .TRUE.
          CALL verify_passwords (npa, passw, npt (nbr), passt (1, nbr),
     &                           active, iline, blabla)
          IF (active) THEN
            found = .TRUE.
          ENDIF
          CALL out (echo, card, first, star, msg, active)
        ELSEIF (.NOT. inside2) THEN
! Level 2
          IF (c7(1:4) == sif) THEN
            star2 = .TRUE.
          ENDIF
          nbr2 = 1
          CALL get_passwords (card (5:), npt2 (nbr2), passt2 (1,nbr2),
     &                        iline, blabla)
          inside2 = .TRUE.
          CALL verify_passwords (npa, passw, npt2(nbr2), passt2(1,nbr2),
     &                           active2, iline, blabla)
          IF (active2) THEN
            found2 = .TRUE.
          ENDIF
          CALL out (echo, card, first, star2, msg, active.AND.active2)
        ELSE
          CALL errmss ('Nested(2) conditionals are not allowed')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          CALL STOP_BY_CODE (0)
        ENDIF
c
      ELSEIF (c7(1:7) == cifnot) THEN
c
c CIFNOT
c
        IF (LEN_TRIM (card) < 8) THEN
          CALL ERRMSS ('No password after CIFNOT')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          CALL STOP_BY_CODE (0)
        ENDIF
        IF (.NOT. inside) THEN
! Level 1
          IF (inside2) THEN
            CALL errmss ('Conditionals are incorrectly nested')
            CALL STOP_BY_CODE (0)
          ENDIF
          nbr = 1
          CALL get_passwords (card (8:), npt (nbr), passt (1,nbr),
     &                        iline, blabla)
          inside = .TRUE.
          negative = .TRUE.
          CALL verify_passwords (npa, passw, npt (nbr), passt (1, nbr),
     &                           active, iline, blabla)
          active = .NOT. active
          IF (active) THEN
            found = .TRUE.
          ENDIF
          CALL out (echo, card, first, star, msg, active)
        ELSEIF (.NOT. inside2) THEN
! Level 2
          nbr2 = 1
          CALL get_passwords (card (8:), npt2 (nbr2), passt2 (1,nbr2),
     &                        iline, blabla)
          inside2 = .TRUE.
          negative2 = .TRUE.
          CALL verify_passwords (npa, passw, npt2(nbr2), passt2(1,nbr2),
     &                           active2, iline, blabla)
          active2 = .NOT. active2
          IF (active2) THEN
```

```
            found2 = .TRUE.
          ENDIF
          CALL out (echo, card, first, star2, msg, active.AND.active2)
        ELSE
          CALL errmss ('Nested(2) conditionals are not allowed')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          CALL STOP_BY_CODE (0)
        ENDIF
c
      ELSEIF (c7(1:6) == celif) THEN
c
c CELIF
c
        IF (LEN_TRIM (card) < 7) THEN
          CALL ERRMSS ('No password after CELIF ')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          CALL STOP_BY_CODE (0)
        ENDIF
        IF (.NOT. inside) THEN
          CALL errmss ('CELIF found while not in conditional')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          CALL STOP_BY_CODE (0)
        ENDIF
        IF (negative .OR. (inside2 .AND. negative2)) THEN
          CALL errmss ('CELIF found while inside CIFNOT')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          CALL STOP_BY_CODE (0)
        ENDIF
        IF (.NOT. inside2) THEN
! Level 1
          nbr = nbr + 1
          IF (nbr > nbran) THEN
            CALL ERRMSS ('Too many branches')
            WRITE (blabla, '("on line", i6)') iline
            CALL errmss (TRIM (blabla))
            CALL STOP_BY_CODE (0)
          ENDIF
          CALL get_passwords (card (7:), npt (nbr), passt (1,nbr),
     &                        iline, blabla)
          CALL compare_passwords (nbr, npt, passt, iline, blabla)
          CALL verify_passwords (npa, passw, npt (nbr), passt (1, nbr),
     &                           active, iline, blabla)
          IF (active) THEN
            found = .TRUE.
          ENDIF
          CALL out (echo, card, first, star, msg, active)
        ELSE
! Level 2
          nbr2 = nbr2 + 1
          IF (nbr2 > nbran) THEN
            CALL ERRMSS ('Too many branches')
            WRITE (blabla, '("on line", i6)') iline
            CALL errmss (TRIM (blabla))
            CALL STOP_BY_CODE (0)
          ENDIF
          CALL get_passwords (card (7:), npt2(nbr2), passt2(1,nbr2),
     &                        iline, blabla)
          CALL compare_passwords (nbr2, npt2, passt2, iline, blabla)
          CALL verify_passwords (npa, passw, npt2(nbr2), passt2(1,nbr2),
     &                           active2, iline, blabla)
          IF (active2) THEN
            found2 = .TRUE.
          ENDIF
          CALL out (echo, card, first, star2, msg, active.AND.active2)
        ENDIF
c
       ELSEIF (c7(1:6) == celse) THEN
c
c CELSE
c
        IF (inside2) THEN
! Level 2
          IF (elsed2) THEN
            CALL ERRMSS ('More than one CELSE found')
            WRITE (blabla, '("on line", i6)') iline
            CALL errmss (TRIM (blabla))
            CALL STOP_BY_CODE (0)
          ELSE
            elsed2 = .TRUE.
          ENDIF
        ELSEIF (inside) THEN
! Level 1
          IF (elsed) THEN
            CALL ERRMSS ('More than one CELSE found')
            WRITE (blabla, '("on line", i6)') iline
            CALL errmss (TRIM (blabla))
            CALL STOP_BY_CODE (0)
          ELSE
            elsed = .TRUE.
          ENDIF
        ENDIF
        !IF (LEN_TRIM (card) > 5) THEN
        !  CALL ERRMSS ('Extra characters after CELSE')
        !  WRITE (blabla, '("on line", i6)') iline
        !  CALL errmss (TRIM (blabla))
        !  CALL STOP_BY_CODE (0)
        !ENDIF
        IF (.NOT. inside) THEN
          CALL errmss ('CELSE found while not in conditional')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          CALL STOP_BY_CODE (0)
        ENDIF
        IF (inside2) THEN
! Level 2
          IF (.NOT. found2) THEN
            active2 = .TRUE.
            found2 = .TRUE.
          ELSE
            active2 = .FALSE.
          ENDIF
          CALL out (echo, card, first, star2, msg, active.AND.active2)
        ELSE
! Level 1
          IF (.NOT. found) THEN
            active = .TRUE.
            found = .TRUE.
          ELSE
            active = .FALSE.
          ENDIF
          CALL out (echo, card, first, star, msg, active)
        ENDIF
c
      ELSEIF (c7(1:7) == cendif) THEN
c
c CENDIF
c
        !IF (LEN_TRIM (card) > 6) THEN
        !  CALL ERRMSS ('Extra characters after CENDIF')
        !  WRITE (blabla, '("on line", i6)') iline
        !  CALL errmss (TRIM (blabla))
        !  CALL STOP_BY_CODE (0)
        !ENDIF
        IF (inside2) THEN
! Level 2
          inside2 = .FALSE.
          nbr2 = 0
          CALL out (echo, card, first, star2, msg, active.AND..TRUE.)
          active2 = .FALSE.
          found2 = .FALSE.
          star2 = .FALSE.
          negative2 = .FALSE.
          elsed2 = .FALSE.
        ELSEIF (inside) THEN
! Level 1
          inside = .FALSE.
          nbr = 0
          CALL out (echo, card, first, star, msg, .TRUE.)
          active = .FALSE.
          found = .FALSE.
          star = .FALSE.
          negative = .FALSE.
          elsed = .FALSE.
        ELSE
          CALL errmss ('CENDIF found while not in conditional')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          CALL STOP_BY_CODE (0)
        ENDIF
c
      ELSE
c
c Text line
c
        IF (inside2) THEN
          IF (active .AND. active2) THEN
            WRITE (6, '(A)') TRIM (card)
          ELSEIF (star2) THEN
            WRITE (6, '(A)') TRIM (first//card(2:))
          ENDIF
        ELSEIF (inside) THEN
          IF (active) THEN
            WRITE (6, '(A)') TRIM (card)
          ELSEIF (star) THEN
            WRITE (6, '(A)') TRIM (first//card(2:))
          ENDIF
        ELSE
          WRITE (6, '(A)') TRIM (card)
        ENDIF
c
      ENDIF
c
      GO TO 1
c
c End of input file
c
  999 IF (inside) THEN
        CALL errmss ('EOF reached while within a CIF construct')
        CALL STOP_BY_CODE (0)
      ENDIF
c
c Return 1 on normal execution (for use in PERL scripts).
c (When an error message is issued, the return code is 0).
c
      CALL STOP_BY_CODE (1)
c
      END PROGRAM epx_filter
c=======================================================================
      SUBROUTINE errmss (msg)
c
      IMPLICIT NONE
c
      CHARACTER (LEN=*) :: msg
c
      WRITE (6,*) "*** ERROR *** "//msg
c
      END SUBROUTINE errmss
c=======================================================================
      SUBROUTINE usage
c
      IMPLICIT NONE
c
      WRITE (6,*) "Usage: epx_filter [-c<character>] mot1 [mot2 ...]"
      WRITE (6,*) "                  <input >output"
c
      END SUBROUTINE usage
c=======================================================================
      SUBROUTINE to_upper (s)
c
      IMPLICIT NONE
c
      CHARACTER (LEN=*), INTENT(INOUT) :: s
c
      INTEGER(4) :: i, k
c
      DO i = 1, LEN_TRIM (s)
        k = ICHAR (s (i:i))
        IF (k >= 97 .and. k <= 122) THEN
          s (i:i) = CHAR (k - 32)
        ENDIF
      END DO
c
```

```
      END SUBROUTINE to_upper
c=======================================================================
      SUBROUTINE get_passwords (card, npt, passt, iline, blabla)
c
      IMPLICIT NONE
c
      INTEGER(4), PARAMETER :: pwdlen = 16 ! Max. length of a password
      INTEGER(4), PARAMETER :: npasst = 8  ! Max. number of passwords
c                                          ! in a line of text file
c
      INTEGER(4), PARAMETER :: crdlen = 132! Max. length of a text 'card'
      CHARACTER (LEN=crdlen) :: c
c
      CHARACTER (LEN=*), INTENT(IN) :: card
      INTEGER(4), INTENT(OUT) :: npt
      CHARACTER (LEN=pwdlen),DIMENSION(npasst), INTENT(OUT) :: passt
      INTEGER(4), INTENT(IN) :: iline
      CHARACTER (LEN=*), INTENT(OUT) :: blabla
c
      INTEGER(4) :: i, k
c
      c = TRIM (card)
      npt = 0
c
      DO
        i = SCAN (TRIM (c), ' ')
        IF (i > 0) THEN
          npt = npt + 1
          IF (npt > npasst) THEN
            CALL errmss ('Too many passwords in text file')
            WRITE (blabla, '("on line", i6)') iline
            CALL errmss (TRIM (blabla))
            CALL STOP_BY_CODE (0)
          ENDIF
          passt (npt) = c (1 : i-1)
          CALL to_upper (passt (npt))
          c = ADJUSTL (c (i+1 :))
        ELSE
          IF (LEN_TRIM (c) > 0) THEN
            npt = npt + 1
            IF (npt > npasst) THEN
              CALL errmss ('Too many passwords in text file')
              WRITE (blabla, '("on line", i6)') iline
              CALL errmss (TRIM (blabla))
              CALL STOP_BY_CODE (0)
            ENDIF
            passt (npt) = TRIM (c)
            CALL to_upper (passt (npt))
          ENDIF
          EXIT
        ENDIF
      END DO
c
      END SUBROUTINE get_passwords
c=======================================================================
      SUBROUTINE compare_passwords (nbr, npt, passt, iline, blabla)
c
      IMPLICIT NONE
c
      INTEGER(4), PARAMETER :: pwdlen = 16 ! Max. length of a password
      INTEGER(4), PARAMETER :: npasst = 8  ! Max. number of passwords
c                                          ! in a line of text file
      INTEGER(4), PARAMETER :: nbran  = 10 ! Max. n. of branches in a CIF
c                                          ! construct
c
      INTEGER(4), INTENT(IN) :: nbr
      INTEGER(4), DIMENSION(nbran), INTENT(IN) :: npt
      CHARACTER (LEN=pwdlen),
     &          DIMENSION(npasst, nbran) :: passt  ! Passwords in each
c                                                  branch of txtfile
      INTEGER(4), INTENT(IN) :: iline
      CHARACTER (LEN=*), INTENT(OUT) :: blabla
c
      CHARACTER (LEN=pwdlen) :: passj, passl
c
      INTEGER(4) :: i, j, k, l
c
      DO i = 2, nbr
        DO j = 1, npt (i)
          passj = passt (j, i)
          DO k = 1, nbr - 1
            DO l = 1, npt (k)
              passl = passt (l, k)
              IF (l /= j .OR. k /= i) THEN
                IF (passj == passl) THEN
                  CALL errmss ('Repeated password ')
                  WRITE (blabla, '(a, "on line", I6)') passj, iline
                  CALL errmss (TRIM (blabla))
                  CALL STOP_BY_CODE (0)
                ENDIF
              ENDIF
            END DO
          END DO
        END DO
      END DO
c
      END SUBROUTINE compare_passwords
c=======================================================================
      SUBROUTINE out (echo, card, first, star, msg, active)
c
      IMPLICIT NONE
c
      INTEGER(4), PARAMETER :: crdlen = 132! Max. length of a text 'card'
c
      LOGICAL(4), INTENT(IN) :: echo
      CHARACTER (LEN=crdlen), INTENT(IN) :: card
      CHARACTER (LEN=1), INTENT(IN) :: first
      LOGICAL(4), INTENT(IN) :: star
      CHARACTER (LEN=21), INTENT(IN) :: msg
      LOGICAL(4), INTENT(IN) :: active
c
      IF (star) THEN
        WRITE (6, '(A)') TRIM (first//card(2:))
      ELSE
        IF (echo) THEN
          WRITE (6, '(A)') TRIM (first//card(2:))
          IF (.NOT. active) THEN
            WRITE (6, '(A)') first//msg
          ENDIF
        ENDIF
      ENDIF
c
      END SUBROUTINE out
c=======================================================================
      SUBROUTINE verify_passwords (npa, passw, npt, passt, active,
     &                             iline, blabla)
c
      IMPLICIT NONE
c
      INTEGER(4), PARAMETER :: pwdlen = 16 ! Max. length of a password
      INTEGER(4), PARAMETER :: npassw = 8  ! Max. number of passwords
c                                          ! given in command line
      INTEGER(4), PARAMETER :: npasst = 8  ! Max. number of passwords
c                                          ! in each line of text file
c
      INTEGER(4), INTENT(IN) :: npa
      CHARACTER (LEN=pwdlen), DIMENSION(npassw), INTENT(IN) :: passw
      INTEGER(4), INTENT(IN) :: npt
      CHARACTER (LEN=pwdlen), DIMENSION(npasst), INTENT(IN) :: passt
      LOGICAL(4), INTENT(INOUT) :: active
      INTEGER(4), INTENT(IN) :: iline
      CHARACTER (LEN=80), INTENT(OUT) :: blabla
c
      INTEGER(4) :: i, j
      LOGICAL(4) :: match
      CHARACTER (LEN=pwdlen) :: passi, passj
c
c Compare current text line passwords with cmdline passwords,
c to see if there is at least one match.
c
      match = .FALSE.
      DO i = 1, npt
        passi = passt (i)
        DO j = 1, npa
          passj = passw (j)
          IF (passj == passi) THEN
            match = .TRUE.
            GO TO 1
          ENDIF
        END DO
      END DO

c
    1 IF (.NOT. active) THEN
c
c If at least one match was found, set active = .TRUE.
c
        IF (match) THEN
          active = .TRUE.
        ENDIF
      ELSE
c
c If a match was found, give error message,
c otherwise set active = .FALSE.
c
        IF (match) THEN
          CALL errmss ('Specified passwords match '//
     &                 'more than one branch')
          WRITE (blabla, '("on line ",I6)') iline
          CALL errmss (TRIM (blabla))
          CALL STOP_BY_CODE (0)
        ELSE
          active = .FALSE.
        ENDIF
      ENDIF
c
      END SUBROUTINE verify_passwords
c=======================================================================
      SUBROUTINE STOP_BY_CODE (stop_code)
      INTEGER, INTENT(IN) :: stop_code
*IF WIN
c ATTENTION! LES LIGNES SUIVANTES DOIVENT ETRE COMMENTEES SOUS AIX
!     IF (stop_code == 0) THEN
!       STOP 0
!     ELSE
!       STOP 1
!     ENDIF
CELSE
c ATTENTION! LES LIGNES SUIVANTES DOIVENT ETRE ACTIVEES SOUS AIX
      IF (stop_code == 0) THEN
        STOP 'FAILED'
      ELSE
        STOP 'OK'
      ENDIF
CENDIF
      END SUBROUTINE STOP_BY_CODE
```

epx_filter_linux.f

```
      PROGRAM epx_filter
!
! New version (fc, 7/10/2004) allowing single-level nesting of conditionals
!
! New version (fc, 7/10/2008) : Correct bug in COMPARE_PASSWORDS, now
!                               one may effectively have up to NBRAN
!                               conditional branches CELIF (as was
!                               initially foreseen), not just one!
!                               Correct bug in VERIFY_PASSWORDS : in case
!                               of error the STOP code is "0" and not "0 1"
!                               (which was returned as 1).
! New version (hb, juillet_09): Syntax A) et C) donne les memes resultats
!                               que le syntax B)
!                               le fichier filtré conserve la meme
!                               longueur de le fichier original
!                               C'est tres utile pour un debogage.
!May 2011                      : ajoutr du flag "%SALOME"
!
!
!
c Filters text files containing the following 3 types of construct:
c
c       Syntax A)             |    Syntax B)             |    Syntax C)
c                             |                          |
c     CIF mot1 [mot2 ...]     |  *IF mot1 [mot2 ...]     |  CIFNOT mot1 [mot2 ...]
```

```
c         [text1]            |     [text1]             |     [text1]
c    [CELIF mot3 [mot4 ...]] | [CELIF mot3 [mot4 ...]] | [CELSE]
c         [text2]            |     [text2]             |     [text2]
c    [CELSE]                 | [CELSE]                 |  CENDIF
c         [text3]            |     [text3]             |
c     CENDIF                 |  CENDIF                 |
c
c Components in [ ] are optional.
c
c The mot1, mot2 etc are passwords given on the command line by the user.
c
c The concatenation of two or more passwords corresponds to a
c logical .OR. between them. Thus CIF mot1 mot2 means IF (mot1 .OR. mot2)
c and CIFNOT mot1 mot2 means IF (.NOT. (mot1 .OR. mot2)).
c
c The CELSE and CENDIF passwords may be optionally followed by a
c comment on the same line. Comments must be separated by at least
c one blank from the preceding keyword. For example:
c CIF toto        <- 'toto' is a password here
c ELSE non-toto   <- 'non-toto' is a comment
c CENDIF toto     <- 'toto' is a comment
c
c Usage: epx_filter [-c<character>] [mot1 [mot2 ...]] <input >output
c
c
c
c For syntaxes A) and C):
c =====================
c
c        -c<character>   This option, if present, causes the use of
c                        the given <character> in place of C in the
c                        output transcription of keywords, and of
c                        the built-in message *** ACCES INTERDIT ***,
c                        that is automatically output in place
c                        of the non-activated text blocks.
c                        Example: '-c%' would give in output %IF,
c                        %ELIF, %ELSE, %ENDIF and %** ACCES INTERDIT ***.
c                        This example may be used for LaTeX documents, that
c                        use % as comment character.
c
c        -c              If no character (i.e. a blank) is specified, then
c                        neither the keyword lines nor the built-in
c                        message are transcribed, so the output file
c                        will contain just the activated text.
c
c For syntax B):
c =============
c
c The non-active branches, if any, are transcribed preceded by the chosen
c (or default) comment character. They remain therefore visible (but inactive)
c in the filtered text.
c
c        -c<character>   This option, if present, causes the use of
c                        the given <character> in place of C in the
c                        output transcription of keywords, and of
c                        (commented) inactive branches.
c
c        -c              NO EFFECT with this syntax
c
!!Stop HB(juillet_09):    Cette partie est obsolete

c Note that:
c =========
c
c    - Keywords CIF, CELIF etc. MUST start in column 1.
c
c    - Keywords are NOT case sensitive, i.e. CIF, cif or CiF are the same.
c
c    - Passwords mot1, mot2 etc. are NOT case sensitive, and may not
c      contain blanks.
c
c    - To avoid ambiguous cases, the filter performs the following tests
c      while filtering the text file:
c
c        Tests on the command line syntax:
c
c          1. The user may not specify twice the same password (or equivalent
c             passwords that differ only by letter case) on the command line.
c
c        Tests on the input text file contents:
c
c          2. Passwords in each branch of a conditional must differ from
c             any passwords in other branches of the same conditional.
c          3. If a user specifies more than one password, all passwords must
c             belong to the same branch of a conditional.
c          4. The CIFNOT syntax does not admit CELIF branches.
c
c    - The program returns 1 if everything was OK, else it returns 0
c
CIF WIN
C ATTENTION! LA LIGNE SUIVANTE DOIT ETRE COMMENTEE SOUS AIX
C     USE dflib ! For GETARG, NARGS
CENDIF
c
      IMPLICIT NONE
c
      INTEGER(4), PARAMETER :: pwdlen = 16 ! Max. length of a password
      INTEGER(4), PARAMETER :: npassw = 8  ! Max. number of passwords
c                                          ! given in command line
      INTEGER(4), PARAMETER :: crdlen = 132! Max. length of a text 'card'
c
      INTEGER(4) :: nar, npa = 0, iline = 0
      INTEGER(4) :: i
CIF WIN
C ATTENTION! LA LIGNE SUIVANTE DOIT ETRE COMMENTEE SOUS AIX
C     INTEGER(2) :: pos, stat
CELSE
c ATTENTION! LA LIGNE SUIVANTE DOIT ETRE ACTIVEE SOUS AIX
      INTEGER(4) :: pos, stat, iargc
CENDIF
      CHARACTER (LEN=pwdlen) :: pwd
      CHARACTER (LEN=pwdlen), DIMENSION(npassw) :: passw  ! Passwords in cmdline
      CHARACTER (LEN=1) :: first = 'C'
      LOGICAL(4) :: echo = .TRUE., inside = .FALSE.
      LOGICAL(4) :: active = .FALSE., found = .FALSE.
      LOGICAL(4) :: star = .FALSE., negative = .FALSE.
      LOGICAL(4) :: elsed = .FALSE.
!
      LOGICAL(4) :: inside2 = .FALSE.
      LOGICAL(4) :: active2 = .FALSE., found2 = .FALSE.
      LOGICAL(4) :: star2 = .FALSE., negative2 = .FALSE.
      LOGICAL(4) :: elsed2 = .FALSE.
c
c echo   : TRUE  if keywords and built-in message are to be transcribed,
c                preceded by the comment character
c          FALSE otherwise
c
c inside : TRUE  if we are inside a CIF or *IF or CIFNOT construct
c          FALSE otherwise
c
c active : TRUE  if we are in the active branch of the construct
c          FALSE otherwise
c
c star   : TRUE  if we are in an *IF construct
c          FALSE otherwise
c
c negative: TRUE  if we are in a CIFNOT construct
c           FALSE otherwise
!
! xxx2 : same meaning as xxx but for the nested conditional
!
! We assume throughout that inside2 implies inside !!!!!!!!!!!!!!!!!!!
c
      CHARACTER (LEN=crdlen) :: card
      CHARACTER (LEN=4) :: cif    = "CIF "
      CHARACTER (LEN=4) :: sif    = "*IF "
      CHARACTER (LEN=7) :: cifnot = "CIFNOT "
      CHARACTER (LEN=6) :: celif  = "CELIF "
      CHARACTER (LEN=6) :: celse  = "CELSE "
      CHARACTER (LEN=7) :: cendif = "CENDIF "
      CHARACTER (LEN=7) :: c7
      CHARACTER (LEN=7) :: csalome= "%SALOME"
c
      INTEGER(4), PARAMETER :: npasst = 9  ! Max. number of passwords
c                                          ! in each line of text file
      INTEGER(4), PARAMETER :: nbran  = 10 ! Max. n. of branches in a CIF
c                                          ! construct
      INTEGER(4), DIMENSION(nbran) :: npt = 0
      INTEGER(4) :: nbr = 0
      CHARACTER (LEN=pwdlen),
     &           DIMENSION(npasst, nbran) :: passt  ! Passwords in each
c                                                    branch of txtfile
! same as above but for the nested conditional
      INTEGER(4), DIMENSION(nbran) :: npt2 = 0
      INTEGER(4) :: nbr2 = 0
      CHARACTER (LEN=pwdlen),
     &           DIMENSION(npasst, nbran) :: passt2 ! Passwords in each
c                                                    branch of txtfile
!
      CHARACTER (LEN=21) :: msg = "** ACCES INTERDIT ***"
c
      CHARACTER (LEN=80) :: blabla
      INTEGER (4) :: ioerror
c
c Check number of arguments
c
CIF WIN
C ATTENTION! LES LIGNES SUIVANTES DOIVENT ETRE COMMENTEES SOUS AIX
C     nar = NARGS() ! Number of command line arguments, including the command
C     IF (nar.lt.1) THEN
CELSE
c ATTENTION! LES LIGNES SUIVANTE DOIVENT ETRE ACTIVEES SOUS AIX
      nar = IARGC()
      IF (nar.lt.0) THEN
CENDIF
        CALL errmss ('WRONG NUMBER OF COMMAND LINE ARGUMENTS')
        CALL usage
        STOP 0
      ENDIF
c
c Retrieve command line option, if any, and passwords (zero or more)
c
CIF WIN
C ATTENTION! LES LIGNES SUIVANTES DOIVENT ETRE COMMENTEES SOUS AIX
C     DO pos = 1, nar-1
C       CALL GETARG (pos, pwd, stat)
CELIF HP-UX
C ATTENTION! LES LIGNES SUIVANTES DOIVENT ETRE COMMENTEES SOUS AIX,
C OSF1, SunOS, IRIX64 MAIS DOIVENT ETRE ACTIVEES SOUS HP-UX
C     DO pos = 1, nar+1
C       CALL GETARG (pos, pwd)
C       stat = LEN_TRIM (pwd)
CELSE
c ATTENTION! LES LIGNES SUIVANTES DOIVENT ETRE ACTIVEES SOUS AIX
      DO pos = 1, nar
        CALL GETARG (pos, pwd)
        stat = LEN_TRIM (pwd)
CENDIF
c
c The GETARG function returns stat=-1 in case of error. If no error,
c pwd contains the retrieved argument and stat its length in characters.
c
        IF (stat <= 0) THEN
          CALL errmss ('Wrong command line argument')
          STOP 0
        ENDIF
        CALL to_upper (pwd)
        IF (stat > 1 .and. pwd (1:2) == '-C') THEN
c
c Option -C<character>
c
          IF (stat == 2) THEN
            echo = .FALSE.
          ELSEIF (stat == 3) THEN
            first = pwd (3:3)
          ELSE
            CALL errmss ('Wrong option -C<character>')
            STOP 0
          ENDIF
        ELSE
c
c Password
c
          npa = npa + 1
          IF (npa > npassw) THEN
            CALL errmss ('Too many passwords')
```

```
            STOP 0
          ENDIF
          CALL to_upper (pwd)
          passw (npa) = pwd
          IF (npa > 1 ) then
            DO i = 1, npa - 1
              IF (passw (npa) == passw (i)) then
                CALL errmss ('The cmdline passwords must be different')
                WRITE (blabla, '("password: ", A)') passw (i)
                CALL errmss (TRIM (blabla))
                STOP 0
              ENDIF
            END DO
          ENDIF
        ENDIF
      END DO
c
c Process input text
c
    1 READ (5, '(A)', IOSTAT=ioerror) card
      IF (ioerror > 0) THEN
c
c Read error encountered
c
        CALL ERRMSS ('While reading')
        WRITE (blabla, '("on line", i6)') iline+1
        CALL errmss (TRIM (blabla))
        STOP 0
      ELSEIF (ioerror < 0) THEN
c
c End-of-file or end-of-record reached
c
        GO TO 999
      ENDIF
      iline = iline + 1
c
      c7 = card (1:7)
      CALL to_upper (c7)
c
c Skip SALOME dedicated commands
      IF (c7(1:7) == csalome) GO TO 1
c
      IF ((c7(1:4) == cif) .OR. (c7(1:4) == sif)) THEN
c
c CIF or *IF
c
        IF (LEN_TRIM (card) < 5) THEN
          CALL ERRMSS ('No password after CIF or *IF')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
        IF (.NOT. inside) THEN
! Level 1
!!hb(juillet_09)          IF (c7(1:4) == sif) THEN
            star = .TRUE.
!!hb(juillet_09)          ENDIF
          IF (inside2) THEN
            CALL errmss ('Conditionals are incorrectly nested')
            STOP 0
          ENDIF
          nbr = 1
          CALL get_passwords (card (5:), npt (nbr), passt (1,nbr),
     &                        iline, blabla)
          inside = .TRUE.
          CALL verify_passwords (npa, passw, npt (nbr), passt (1, nbr),
     &                           active, iline, blabla)
          IF (active) THEN
            found = .TRUE.
          ENDIF
          CALL out (echo, card, first, star, msg, active)
        ELSEIF (.NOT. inside2) THEN
! Level 2
!!hb(juillet_09)          IF (c7(1:4) == sif) THEN
            star2 = .TRUE.
!!hb(juillet_09)          ENDIF
          nbr2 = 1
          CALL get_passwords (card (5:), npt2 (nbr2), passt2 (1,nbr2),
     &                        iline, blabla)
          inside2 = .TRUE.
          CALL verify_passwords (npa, passw, npt2(nbr2), passt2(1,nbr2),
     &                           active2, iline, blabla)
          IF (active2) THEN
            found2 = .TRUE.
          ENDIF
          CALL out (echo, card, first, star2, msg, active.AND.active2)
        ELSE
          CALL errmss ('Nested(2) conditionals are not allowed')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
c
      ELSEIF (c7(1:7) == cifnot) THEN
c
c CIFNOT
c
        IF (LEN_TRIM (card) < 8) THEN
          CALL ERRMSS ('No password after CIFNOT')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
        IF (.NOT. inside) THEN
! Level 1
          IF (inside2) THEN
            CALL errmss ('Conditionals are incorrectly nested')
            STOP 0
          ENDIF
          nbr = 1
          CALL get_passwords (card (8:), npt (nbr), passt (1,nbr),
     &                        iline, blabla)
          inside = .TRUE.
          negative = .TRUE.
          CALL verify_passwords (npa, passw, npt (nbr), passt (1, nbr),
     &                           active, iline, blabla)
          active = .NOT. active
          IF (active) THEN
            found = .TRUE.
          ENDIF
          CALL out (echo, card, first, star, msg, active)
        ELSEIF (.NOT. inside2) THEN
! Level 2
          nbr2 = 1
          CALL get_passwords (card (8:), npt2 (nbr2), passt2 (1,nbr2),
     &                        iline, blabla)
          inside2 = .TRUE.
          negative2 = .TRUE.
          CALL verify_passwords (npa, passw, npt2(nbr2), passt2(1,nbr2),
     &                           active2, iline, blabla)
          active2 = .NOT. active2
          IF (active2) THEN
            found2 = .TRUE.
          ENDIF
          CALL out (echo, card, first, star2, msg, active.AND.active2)
        ELSE
          CALL errmss ('Nested(2) conditionals are not allowed')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
c
      ELSEIF (c7(1:6) == celif) THEN
c
c CELIF
c
        IF (LEN_TRIM (card) < 7) THEN
          CALL ERRMSS ('No password after CELIF ')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
        IF (.NOT. inside) THEN
          CALL errmss ('CELIF found while not in conditional')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
        IF (negative .OR. (inside2 .AND. negative2)) THEN
          CALL errmss ('CELIF found while inside CIFNOT')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
        IF (.NOT. inside2) THEN
! Level 1
          nbr = nbr + 1
          IF (nbr > nbran) THEN
            CALL ERRMSS ('Too many branches')
            WRITE (blabla, '("on line", i6)') iline
            CALL errmss (TRIM (blabla))
            STOP 0
          ENDIF
          CALL get_passwords (card (7:), npt (nbr), passt (1,nbr),
     &                        iline, blabla)
          CALL compare_passwords (nbr, npt, passt, iline, blabla)
          CALL verify_passwords (npa, passw, npt (nbr), passt (1, nbr),
     &                           active, iline, blabla)
          IF (active) THEN
            found = .TRUE.
          ENDIF
          CALL out (echo, card, first, star, msg, active)
        ELSE
! Level 2
          nbr2 = nbr2 + 1
          IF (nbr2 > nbran) THEN
            CALL ERRMSS ('Too many branches')
            WRITE (blabla, '("on line", i6)') iline
            CALL errmss (TRIM (blabla))
            STOP 0
          ENDIF
          CALL get_passwords (card (7:), npt2(nbr2), passt2(1,nbr2),
     &                        iline, blabla)
          CALL compare_passwords (nbr2, npt2, passt2, iline, blabla)
          CALL verify_passwords (npa, passw, npt2(nbr2), passt2(1,nbr2),
     &                           active2, iline, blabla)
          IF (active2) THEN
            found2 = .TRUE.
          ENDIF
          CALL out (echo, card, first, star2, msg, active.AND.active2)
        ENDIF
c
      ELSEIF (c7(1:6) == celse) THEN
c
c CELSE
c
        IF (inside2) THEN
! Level 2
          IF (elsed2) THEN
            CALL ERRMSS ('More than one CELSE found')
            WRITE (blabla, '("on line", i6)') iline
            CALL errmss (TRIM (blabla))
            STOP 0
          ELSE
            elsed2 = .TRUE.
          ENDIF
        ELSEIF (inside) THEN
! Level 1
          IF (elsed) THEN
            CALL ERRMSS ('More than one CELSE found')
            WRITE (blabla, '("on line", i6)') iline
            CALL errmss (TRIM (blabla))
            STOP 0
          ELSE
            elsed = .TRUE.
          ENDIF
        ENDIF
        !IF (LEN_TRIM (card) > 5) THEN
        !  CALL ERRMSS ('Extra characters after CELSE')
        !  WRITE (blabla, '("on line", i6)') iline
        !  CALL errmss (TRIM (blabla))
        !  STOP 0
        !ENDIF
        IF (.NOT. inside) THEN
          CALL errmss ('CELSE found while not in conditional')
          WRITE (blabla, '("on line", i6)') iline
```

```
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
        IF (inside2) THEN
! Level 2
          IF (.NOT. found2) THEN
            active2 = .TRUE.
            found2 = .TRUE.
          ELSE
            active2 = .FALSE.
          ENDIF
          CALL out (echo, card, first, star2, msg, active.AND.active2)
        ELSE
! Level 1
          IF (.NOT. found) THEN
            active = .TRUE.
            found = .TRUE.
          ELSE
            active = .FALSE.
          ENDIF
          CALL out (echo, card, first, star, msg, active)
        ENDIF
c
      ELSEIF (c7(1:7) == cendif) THEN
c
c CENDIF
c
        !IF (LEN_TRIM (card) > 6) THEN
        !  CALL ERRMSS ('Extra characters after CENDIF')
        !  WRITE (blabla, '("on line", i6)') iline
        !  CALL errmss (TRIM (blabla))
        !  STOP 0
        !ENDIF
        IF (inside2) THEN
! Level 2
          inside2 = .FALSE.
          nbr2 = 0
          CALL out (echo, card, first, star2, msg, active.AND..TRUE.)
          active2 = .FALSE.
          found2 = .FALSE.
          star2 = .FALSE.
          negative2 = .FALSE.
          elsed2 = .FALSE.
        ELSEIF (inside) THEN
! Level 1
          inside = .FALSE.
          nbr = 0
          CALL out (echo, card, first, star, msg, .TRUE.)
          active = .FALSE.
          found = .FALSE.
          star = .FALSE.
          negative = .FALSE.
          elsed = .FALSE.
        ELSE
          CALL errmss ('CENDIF found while not in conditional')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
c
      ELSE
c
c Text line
c
        IF (inside2) THEN
          IF (active .AND. active2) THEN
            WRITE (6, '(A)') TRIM (card)
          ELSEIF (star2 .and. echo) THEN
            WRITE (6, '(A)') TRIM (first//card(2:))
          ENDIF
        ELSEIF (inside) THEN
          IF (active) THEN
            WRITE (6, '(A)') TRIM (card)
          ELSEIF (star .and. echo) THEN
            WRITE (6, '(A)') TRIM (first//card(2:))
          ENDIF
        ELSE
          WRITE (6, '(A)') TRIM (card)
        ENDIF
c
      ENDIF
c
      GO TO 1
c
c End of input file
c
  999 IF (inside) THEN
        CALL errmss ('EOF reached while within a CIF construct')
        STOP 0
      ENDIF
c
c Return 1 on normal execution (for use in PERL scripts).
c (When an error message is issued, the return code is 0).
c
      STOP 1
c
      END PROGRAM epx_filter
c=======================================================================
      SUBROUTINE errmss (msg)
c
      IMPLICIT NONE
c
      CHARACTER (LEN=*) :: msg
c
      WRITE (6,*) "*** ERROR *** "//msg
c
      END SUBROUTINE errmss
c=======================================================================
      SUBROUTINE usage
c
      IMPLICIT NONE
c
      WRITE (6,*) "Usage: epx_filter [-c<character>] mot1 [mot2 ...]"
      WRITE (6,*) "                  <input >output"
c
      END SUBROUTINE usage
c=======================================================================
      SUBROUTINE to_upper (s)
c
      IMPLICIT NONE
c
      CHARACTER (LEN=*), INTENT(INOUT) :: s
c
      INTEGER(4) :: i, k
c
      DO i = 1, LEN_TRIM (s)
        k = ICHAR (s (i:i))
        IF (k >= 97 .and. k <= 122) THEN
          s (i:i) = CHAR (k - 32)
        ENDIF
      END DO
c
      END SUBROUTINE to_upper
c=======================================================================
      SUBROUTINE get_passwords (card, npt, passt, iline, blabla)
c
      IMPLICIT NONE
c
      INTEGER(4), PARAMETER :: pwdlen = 16 ! Max. length of a password
      INTEGER(4), PARAMETER :: npasst = 9  ! Max. number of passwords
c                                          ! in a line of text file
c
      INTEGER(4), PARAMETER :: crdlen = 132! Max. length of a text 'card'
      CHARACTER (LEN=crdlen) :: c
c
      CHARACTER (LEN=*), INTENT(IN) :: card
      INTEGER(4), INTENT(OUT) :: npt
      CHARACTER (LEN=pwdlen),DIMENSION(npasst), INTENT(OUT) :: passt
      INTEGER(4), INTENT(IN) :: iline
      CHARACTER (LEN=*), INTENT(OUT) :: blabla
c
      INTEGER(4) :: i, k
c
      c = TRIM (card)
      npt = 0
c
      DO
        i = SCAN (TRIM (c), ' ')
        IF (i > 0) THEN
          npt = npt + 1
          IF (npt > npasst) THEN
            CALL errmss ('Too many passwords in text file')
            WRITE (blabla, '("on line", i6)') iline
            CALL errmss (TRIM (blabla))
            STOP 0
          ENDIF
          passt (npt) = c (1 : i-1)
          CALL to_upper (passt (npt))
          c = ADJUSTL (c (i+1 :))
        ELSE
          IF (LEN_TRIM (c) > 0) THEN
            npt = npt + 1
            IF (npt > npasst) THEN
              CALL errmss ('Too many passwords in text file')
              WRITE (blabla, '("on line", i6)') iline
              CALL errmss (TRIM (blabla))
              STOP 0
            ENDIF
            passt (npt) = TRIM (c)
            CALL to_upper (passt (npt))
          ENDIF
          EXIT
        ENDIF
      END DO
c
      END SUBROUTINE get_passwords
c=======================================================================
      SUBROUTINE compare_passwords (nbr, npt, passt, iline, blabla)
c
      IMPLICIT NONE
c
      INTEGER(4), PARAMETER :: pwdlen = 16 ! Max. length of a password
      INTEGER(4), PARAMETER :: npasst = 8  ! Max. number of passwords
c                                          ! in a line of text file
      INTEGER(4), PARAMETER :: nbran  = 10 ! Max. n. of branches in a CIF
c                                          ! construct
c
      INTEGER(4), INTENT(IN) :: nbr
      INTEGER(4), DIMENSION(nbran), INTENT(IN) :: npt
      CHARACTER (LEN=pwdlen),
     &           DIMENSION(npasst, nbran) :: passt  ! Passwords in each
c                                                    branch of txtfile
      INTEGER(4), INTENT(IN) :: iline
      CHARACTER (LEN=*), INTENT(OUT) :: blabla
c
      CHARACTER (LEN=pwdlen) :: passj, passl
c
      INTEGER(4) :: i, j, k, l
c
      DO i = 2, nbr
        DO j = 1, npt (i)
          passj = passt (j, i)
          DO k = 1, nbr - 1
            DO l = 1, npt (k)
              passl = passt (l, k)
              IF (l /= j .OR. k /= i) THEN
                IF (passj == passl) THEN
                  CALL errmss ('Repeated password ')
                  WRITE (blabla, '(a, "on line", I6)') passj, iline
                  CALL errmss (TRIM (blabla))
                  STOP 0
                ENDIF
              ENDIF
            END DO
          END DO
        END DO
      END DO
c
      END SUBROUTINE compare_passwords
c=======================================================================
      SUBROUTINE out (echo, card, first, star, msg, active)
c
      IMPLICIT NONE
c
      INTEGER(4), PARAMETER :: crdlen = 132! Max. length of a text 'card'
c
```

```
      LOGICAL(4), INTENT(IN) :: echo
      CHARACTER (LEN=crdlen), INTENT(IN) :: card
      CHARACTER (LEN=1), INTENT(IN) :: first
      LOGICAL(4), INTENT(IN) :: star
      CHARACTER (LEN=21), INTENT(IN) :: msg
      LOGICAL(4), INTENT(IN) :: active
c
      IF (star) THEN
        IF (echo) WRITE (6, '(A)') TRIM (first//card(2:))
      ELSE
        IF (echo) THEN
          WRITE (6, '(A)') TRIM (first//card(2:))
          IF (.NOT. active) THEN
            WRITE (6, '(A)') first//msg
          ENDIF
        ENDIF
      ENDIF
c
      END SUBROUTINE out
c=======================================================================
      SUBROUTINE verify_passwords (npa, passw, npt, passt, active,
     &                             iline, blabla)
c
      IMPLICIT NONE
c
      INTEGER(4), PARAMETER :: pwdlen = 16 ! Max. length of a password
      INTEGER(4), PARAMETER :: npassw = 8  ! Max. number of passwords
c                                          ! given in command line
      INTEGER(4), PARAMETER :: npasst = 8  ! Max. number of passwords
c                                          ! in each line of text file
c
      INTEGER(4), INTENT(IN) :: npa
      CHARACTER (LEN=pwdlen), DIMENSION(npassw), INTENT(IN) :: passw
      INTEGER(4), INTENT(IN) :: npt
      CHARACTER (LEN=pwdlen), DIMENSION(npasst), INTENT(IN) :: passt
      LOGICAL(4), INTENT(INOUT) :: active
      INTEGER(4), INTENT(IN) :: iline
      CHARACTER (LEN=80), INTENT(OUT) :: blabla
c
      INTEGER(4) :: i, j
      LOGICAL(4) :: match
      CHARACTER (LEN=pwdlen) :: passi, passj
c
c Compare current text line passwords with cmdline passwords,
c to see if there is at least one match.
c
      match = .FALSE.
      DO i = 1, npt
        passi = passt (i)
        DO j = 1, npa
          passj = passw (j)
          IF (passj == passi) THEN
            match = .TRUE.
            GO TO 1
          ENDIF
        END DO
      END DO

c
    1 IF (.NOT. active) THEN
c
c If at least one match was found, set active = .TRUE.
c
        IF (match) THEN
          active = .TRUE.
        ENDIF
      ELSE
c
c If a match was found, give error message,
c otherwise set active = .FALSE.
c
        IF (match) THEN
          CALL errmss ('Specified passwords match '//
     &                 'more than one branch')
          WRITE (blabla, '("on line ",I6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ELSE
          active = .FALSE.
        ENDIF
      ENDIF
c
      END SUBROUTINE verify_passwords
```

epx_filter_manual.pl

```
#-------------------------------------------------------------------------------
# Filter LaTeX source for the EUROPLEXUS manual by epx_filter
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$errfil = "epx_filter_manual.err";
$keys = "WIN QWIN";
$keys2 = " ";
$blan = " ";
#-------------------------------------------------------------------------------
# Check number of arguments, must be at least 1
#
if ($#ARGV < 0) {                       # Index of last argument must be >= 0
  $errmsg = "Usage: epx_filter_manual [-q] [-k KEY] [-x] name(s)[.ttx]\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#       -q       quiet (do not print subsequent switch echo)
#       -k KEY   add key KEY to filtering keys
#       -x       suppress all default filtering keys
#
while ($ARGV[0] =~ /^-/) {
  $_ = shift;
  if (/^-q(.*)/) {
    $quiet = "yes";
  }
  elsif (/^-k$/) {
    $_ = shift;
    $keys2 = $keys2.$_.$blan;
    if (! defined ($quiet)) {
      printf "Command line switch: additional filter keys $keys2\n";
    }
  }
  elsif (/^-x(.*)/) {
    printf "Command line switch: suppress default filter keys $keys\n";
    $keys = " ";
  }
  else {
    $errmsg = "ERROR: unknown switch: $_\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
}
#-------------------------------------------------------------------------------
#
$keys = $keys.$keys2;
#-------------------------------------------------------------------------------
# All remaining arguments are file names: do the globbing
#
@FileNames = glob (join ($",@ARGV));
#-------------------------------------------------------------------------------
# Loop on file(s) to be compiled
#
foreach $name (@FileNames) {
#
  $base = $name;
  $base =~ s/.ttx//;
  $infile = "$base.ttx";
  $outfile = "$base.tex";
  $errfile = "$base.err";
#-------------------------------------------------------------------------------
# Verify that input file is present
#
  if (! -r $infile) {
    $errmsg =  "File $infile not found!\007\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
#-------------------------------------------------------------------------------
# Process .ttx file to obtain .tex file
#
  open (ERR, "> $errfile");
  print ERR "Filtering manual file $infile with keys: $keys\n";
  if (! defined ($quiet)) {
    print "Filtering manual file $infile with keys: $keys\n";
  }
#
  if (! system("epx_filter -c  $keys <$infile >$outfile 2>_devnull")) {
    $errmsg = "Filtering ERROR(S) in $infile! (See $errfile & $outfile)\007\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
#
  close ERR;
}
#
exit;


#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_filter_win.f

```
      PROGRAM epx_filter
!
! New version (fc, 7/10/2004) allowing single-level nesting of conditionals
!
! New version (fc, 7/10/2008) : Correct bug in COMPARE_PASSWORDS, now
!                               one may effectively have up to NBRAN
!                               conditional branches CELIF (as was
!                               initially foreseen), not just one!
!                               Correct bug in VERIFY_PASSWORDS : in case
!                               of error the STOP code is "0" and not "0 1"
!                               (which was returned as 1).
! New version (hb, juillet_09): Syntax A) et C) donne les memes resultats
!                               que le syntax B)
!                               le fichier filtré conserve la meme
!                               longueur de le fichier original
!                               C'est tres utile pour un debogage.
!May 2011                      : ajoutr du flag "%SALOME"
!
!
!
c Filters text files containing the following 3 types of construct:
c
c       Syntax A)             |    Syntax B)             |    Syntax C)
c                             |                          |
c     CIF mot1 [mot2 ...]     |  *IF mot1 [mot2 ...]     |  CIFNOT mot1 [mot2 ...]
c          [text1]            |     [text1]              |     [text1]
c    [CELIF mot3 [mot4 ...]] | [CELIF mot3 [mot4 ...]] | [CELSE]
c          [text2]            |     [text2]              |     [text2]
c    [CELSE]                  | [CELSE]                  |  CENDIF
c          [text3]            |     [text3]              |
c     CENDIF                  |  CENDIF                  |
c
c Components in [ ] are optional.
c
c The mot1, mot2 etc are passwords given on the command line by the user.
c
c The concatenation of two or more passwords corresponds to a
c logical .OR. between them. Thus CIF mot1 mot2 means IF (mot1 .OR. mot2)
c and CIFNOT mot1 mot2 means IF (.NOT. (mot1 .OR. mot2)).
c
c The CELSE and CENDIF passwords may be optionally followed by a
c comment on the same line. Comments must be separated by at least
c one blank from the preceding keyword. For example:
```

```
c CIF toto        <- 'toto' is a password here
c ELSE non-toto   <- 'non-toto' is a comment
c CENDIF toto     <- 'toto' is a comment
c
c Usage: epx_filter [-c<character>] [mot1 [mot2 ...]] <input >output
c
c
c
c For syntaxes A) and C):
c =====================
c
c        -c<character>   This option, if present, causes the use of
c                        the given <character> in place of C in the
c                        output transcription of keywords, and of
c                        the built-in message *** ACCES INTERDIT ***,
c                        that is automatically output in place
c                        of the non-activated text blocks.
c                        Example: '-c%' would give in output %IF,
c                        %ELIF, %ELSE, %ENDIF and %** ACCES INTERDIT ***.
c                        This example may be used for LaTeX documents, that
c                        use % as comment character.
c
c        -c              If no character (i.e. a blank) is specified, then
c                        neither the keyword lines nor the built-in
c                        message are transcribed, so the output file
c                        will contain just the activated text.
c
c For syntax B):
c =============
c
c The non-active branches, if any, are transcribed preceded by the chosen
c (or default) comment character. They remain therefore visible (but inactive)
c in the filtered text.
c
c        -c<character>   This option, if present, causes the use of
c                        the given <character> in place of C in the
c                        output transcription of keywords, and of
c                        (commented) inactive branches.
c
c        -c              NO EFFECT with this syntax
c
!!Stop HB(juillet_09):    Cette partie est obsolete

c Note that:
c =========
c
c    - Keywords CIF, CELIF etc. MUST start in column 1.
c
c    - Keywords are NOT case sensitive, i.e. CIF, cif or CiF are the same.
c
c    - Passwords mot1, mot2 etc. are NOT case sensitive, and may not
c      contain blanks.
c
c    - To avoid ambiguous cases, the filter performs the following tests
c      while filtering the text file:
c
c        Tests on the command line syntax:
c
c          1. The user may not specify twice the same password (or equivalent
c             passwords that differ only by letter case) on the command line.
c
c        Tests on the input text file contents:
c
c          2. Passwords in each branch of a conditional must differ from
c             any passwords in other branches of the same conditional.
c          3. If a user specifies more than one password, all passwords must
c             belong to the same branch of a conditional.
c          4. The CIFNOT syntax does not admit CELIF branches.
c
c    - The program returns 1 if everything was OK, else it returns 0
c
CIF WIN
c ATTENTION! LA LIGNE SUIVANTE DOIT ETRE COMMENTEE SOUS AIX
      USE dflib ! For GETARG, NARGS
CENDIF
c
      IMPLICIT NONE
c
      INTEGER(4), PARAMETER :: pwdlen = 16 ! Max. length of a password
      INTEGER(4), PARAMETER :: npassw = 8  ! Max. number of passwords
c                                          ! given in command line
      INTEGER(4), PARAMETER :: crdlen = 132! Max. length of a text 'card'
c
      INTEGER(4) :: nar, npa = 0, iline = 0
      INTEGER(4) :: i
CIF WIN
c ATTENTION! LA LIGNE SUIVANTE DOIT ETRE COMMENTEE SOUS AIX
      INTEGER(2) :: pos, stat
CELSE
C ATTENTION! LA LIGNE SUIVANTE DOIT ETRE ACTIVEE SOUS AIX
C     INTEGER(4) :: pos, stat, iargc
CENDIF
      CHARACTER (LEN=pwdlen) :: pwd
      CHARACTER (LEN=pwdlen), DIMENSION(npassw) :: passw  ! Passwords in cmdline
      CHARACTER (LEN=1) :: first = 'C'
      LOGICAL(4) :: echo = .TRUE., inside = .FALSE.
      LOGICAL(4) :: active = .FALSE., found = .FALSE.
      LOGICAL(4) :: star = .FALSE., negative = .FALSE.
      LOGICAL(4) :: elsed = .FALSE.
!
      LOGICAL(4) :: inside2 = .FALSE.
      LOGICAL(4) :: active2 = .FALSE., found2 = .FALSE.
      LOGICAL(4) :: star2 = .FALSE., negative2 = .FALSE.
      LOGICAL(4) :: elsed2 = .FALSE.
c
c echo   : TRUE  if keywords and built-in message are to be transcribed,
c                preceded by the comment character
c          FALSE otherwise
c
c inside : TRUE  if we are inside a CIF or *IF or CIFNOT construct
c          FALSE otherwise
c
c active : TRUE  if we are in the active branch of the construct
c          FALSE otherwise
c
c star   : TRUE  if we are in an *IF construct
c          FALSE otherwise
c
c negative: TRUE  if we are in a CIFNOT construct
c           FALSE otherwise
!
! xxx2 : same meaning as xxx but for the nested conditional
!
! We assume throughout that inside2 implies inside !!!!!!!!!!!!!!!!!!!!
c
      CHARACTER (LEN=crdlen) :: card
      CHARACTER (LEN=4) :: cif    = "CIF "
      CHARACTER (LEN=4) :: sif    = "*IF "
      CHARACTER (LEN=7) :: cifnot = "CIFNOT "
      CHARACTER (LEN=6) :: celif  = "CELIF "
      CHARACTER (LEN=6) :: celse  = "CELSE "
      CHARACTER (LEN=7) :: cendif = "CENDIF "
      CHARACTER (LEN=7) :: c7
      CHARACTER (LEN=7) :: csalome= "%SALOME"
c
      INTEGER(4), PARAMETER :: npasst = 9  ! Max. number of passwords
c                                          ! in each line of text file
      INTEGER(4), PARAMETER :: nbran  = 10 ! Max. n. of branches in a CIF
c                                          ! construct
      INTEGER(4), DIMENSION(nbran) :: npt = 0
      INTEGER(4) :: nbr = 0
      CHARACTER (LEN=pwdlen),
     &           DIMENSION(npasst, nbran) :: passt  ! Passwords in each
c                                                   branch of txtfile
! same as above but for the nested conditional
      INTEGER(4), DIMENSION(nbran) :: npt2 = 0
      INTEGER(4) :: nbr2 = 0
      CHARACTER (LEN=pwdlen),
     &           DIMENSION(npasst, nbran) :: passt2 ! Passwords in each
c                                                   branch of txtfile
!
      CHARACTER (LEN=21) :: msg = "** ACCES INTERDIT ***"
c
      CHARACTER (LEN=80) :: blabla
      INTEGER (4) :: ioerror
c
c Check number of arguments
c
CIF WIN
c ATTENTION! LES LIGNES SUIVANTES DOIVENT ETRE COMMENTEES SOUS AIX
      nar = NARGS() ! Number of command line arguments, including the command
      IF (nar.lt.1) THEN
CELSE
C ATTENTION! LES LIGNES SUIVANTE DOIVENT ETRE ACTIVEES SOUS AIX
C     nar = IARGC()
C     IF (nar.lt.0) THEN
CENDIF
        CALL errmss ('WRONG NUMBER OF COMMAND LINE ARGUMENTS')
        CALL usage
        STOP 0
      ENDIF
c
c Retrieve command line option, if any, and passwords (zero or more)
c
CIF WIN
c ATTENTION! LES LIGNES SUIVANTES DOIVENT ETRE COMMENTEES SOUS AIX
      DO pos = 1, nar-1
        CALL GETARG (pos, pwd, stat)
CELIF HP-UX
C ATTENTION! LES LIGNES SUIVANTES DOIVENT ETRE COMMENTEES SOUS AIX,
C OSF1, SunOS, IRIX64 MAIS DOIVENT ETRE ACTIVEES SOUS HP-UX
C     DO pos = 1, nar+1
C       CALL GETARG (pos, pwd)
C       stat = LEN_TRIM (pwd)
CELSE
C ATTENTION! LES LIGNES SUIVANTES DOIVENT ETRE ACTIVEES SOUS AIX
C     DO pos = 1, nar
C       CALL GETARG (pos, pwd)
C       stat = LEN_TRIM (pwd)
CENDIF
c
c The GETARG function returns stat=-1 in case of error. If no error,
c pwd contains the retrieved argument and stat its length in characters.
c
        IF (stat <= 0) THEN
          CALL errmss ('Wrong command line argument')
          STOP 0
        ENDIF
        CALL to_upper (pwd)
        IF (stat > 1 .and. pwd (1:2) == '-C') THEN
c
c Option -C<character>
c
          IF (stat == 2) THEN
            echo = .FALSE.
          ELSEIF (stat == 3) THEN
            first = pwd (3:3)
          ELSE
            CALL errmss ('Wrong option -C<character>')
            STOP 0
          ENDIF
        ELSE
c
c Password
c
          npa = npa + 1
          IF (npa > npassw) THEN
            CALL errmss ('Too many passwords')
            STOP 0
          ENDIF
          CALL to_upper (pwd)
          passw (npa) = pwd
          IF (npa > 1 ) then
            DO i = 1, npa - 1
              IF (passw (npa) == passw (i)) then
                CALL errmss ('The cmdline passwords must be different')
                WRITE (blabla, '("password: ", A)') passw (i)
                CALL errmss (TRIM (blabla))
                STOP 0
              ENDIF
            END DO
          ENDIF
        ENDIF
      END DO
c
c Process input text
```

```
c
    1 READ (5, '(A)', IOSTAT=ioerror) card
      IF (ioerror > 0) THEN
c
c Read error encountered
c
        CALL ERRMSS ('While reading')
        WRITE (blabla, '("on line", i6)') iline+1
        CALL errmss (TRIM (blabla))
        STOP 0
      ELSEIF (ioerror < 0) THEN
c
c End-of-file or end-of-record reached
c
        GO TO 999
      ENDIF
      iline = iline + 1
c
      c7 = card (1:7)
      CALL to_upper (c7)
c
c Skip SALOME dedicated commands
      IF (c7(1:7) == csalome) GO TO 1
c
      IF ((c7(1:4) == cif) .OR. (c7(1:4) == sif)) THEN
c
c CIF or *IF
c
        IF (LEN_TRIM (card) < 5) THEN
          CALL ERRMSS ('No password after CIF or *IF')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
        IF (.NOT. inside) THEN
! Level 1
!!hb(juillet_09)          IF (c7(1:4) == sif) THEN
            star = .TRUE.
!!hb(juillet_09)          ENDIF
          IF (inside2) THEN
            CALL errmss ('Conditionals are incorrectly nested')
            STOP 0
          ENDIF
          nbr = 1
          CALL get_passwords (card (5:), npt (nbr), passt (1,nbr),
     &                        iline, blabla)
          inside = .TRUE.
          CALL verify_passwords (npa, passw, npt (nbr), passt (1, nbr),
     &                           active, iline, blabla)
          IF (active) THEN
            found = .TRUE.
          ENDIF
          CALL out (echo, card, first, star, msg, active)
        ELSEIF (.NOT. inside2) THEN
! Level 2
!!hb(juillet_09)          IF (c7(1:4) == sif) THEN
            star2 = .TRUE.
!!hb(juillet_09)          ENDIF
          nbr2 = 1
          CALL get_passwords (card (5:), npt2 (nbr2), passt2 (1,nbr2),
     &                        iline, blabla)
          inside2 = .TRUE.
          CALL verify_passwords (npa, passw, npt2(nbr2), passt2(1,nbr2),
     &                           active2, iline, blabla)
          IF (active2) THEN
            found2 = .TRUE.
          ENDIF
          CALL out (echo, card, first, star2, msg, active.AND.active2)
        ELSE
          CALL errmss ('Nested(2) conditionals are not allowed')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
c
      ELSEIF (c7(1:7) == cifnot) THEN
c
c CIFNOT
c
        IF (LEN_TRIM (card) < 8) THEN
          CALL ERRMSS ('No password after CIFNOT')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
        IF (.NOT. inside) THEN
! Level 1
          IF (inside2) THEN
            CALL errmss ('Conditionals are incorrectly nested')
            STOP 0
          ENDIF
          nbr = 1
          CALL get_passwords (card (8:), npt (nbr), passt (1,nbr),
     &                        iline, blabla)
          inside = .TRUE.
          negative = .TRUE.
          CALL verify_passwords (npa, passw, npt (nbr), passt (1, nbr),
     &                           active, iline, blabla)
          active = .NOT. active
          IF (active) THEN
            found = .TRUE.
          ENDIF
          CALL out (echo, card, first, star, msg, active)
        ELSEIF (.NOT. inside2) THEN
! Level 2
          nbr2 = 1
          CALL get_passwords (card (8:), npt2 (nbr2), passt2 (1,nbr2),
     &                        iline, blabla)
          inside2 = .TRUE.
          negative2 = .TRUE.
          CALL verify_passwords (npa, passw, npt2(nbr2), passt2(1,nbr2),
     &                           active2, iline, blabla)
          active2 = .NOT. active2
          IF (active2) THEN
            found2 = .TRUE.
          ENDIF
          CALL out (echo, card, first, star2, msg, active.AND.active2)
        ELSE
          CALL errmss ('Nested(2) conditionals are not allowed')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
c
      ELSEIF (c7(1:6) == celif) THEN
c
c CELIF
c
        IF (LEN_TRIM (card) < 7) THEN
          CALL ERRMSS ('No password after CELIF ')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
        IF (.NOT. inside) THEN
          CALL errmss ('CELIF found while not in conditional')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
        IF (negative .OR. (inside2 .AND. negative2)) THEN
          CALL errmss ('CELIF found while inside CIFNOT')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
        IF (.NOT. inside2) THEN
! Level 1
          nbr = nbr + 1
          IF (nbr > nbran) THEN
            CALL ERRMSS ('Too many branches')
            WRITE (blabla, '("on line", i6)') iline
            CALL errmss (TRIM (blabla))
            STOP 0
          ENDIF
          CALL get_passwords (card (7:), npt (nbr), passt (1,nbr),
     &                        iline, blabla)
          CALL compare_passwords (nbr, npt, passt, iline, blabla)
          CALL verify_passwords (npa, passw, npt (nbr), passt (1, nbr),
     &                           active, iline, blabla)
          IF (active) THEN
            found = .TRUE.
          ENDIF
          CALL out (echo, card, first, star, msg, active)
        ELSE
! Level 2
          nbr2 = nbr2 + 1
          IF (nbr2 > nbran) THEN
            CALL ERRMSS ('Too many branches')
            WRITE (blabla, '("on line", i6)') iline
            CALL errmss (TRIM (blabla))
            STOP 0
          ENDIF
          CALL get_passwords (card (7:), npt2(nbr2), passt2(1,nbr2),
     &                        iline, blabla)
          CALL compare_passwords (nbr2, npt2, passt2, iline, blabla)
          CALL verify_passwords (npa, passw, npt2(nbr2), passt2(1,nbr2),
     &                           active2, iline, blabla)
          IF (active2) THEN
            found2 = .TRUE.
          ENDIF
          CALL out (echo, card, first, star2, msg, active.AND.active2)
        ENDIF
c
      ELSEIF (c7(1:6) == celse) THEN
c
c CELSE
c
        IF (inside2) THEN
! Level 2
          IF (elsed2) THEN
            CALL ERRMSS ('More than one CELSE found')
            WRITE (blabla, '("on line", i6)') iline
            CALL errmss (TRIM (blabla))
            STOP 0
          ELSE
            elsed2 = .TRUE.
          ENDIF
        ELSEIF (inside) THEN
! Level 1
          IF (elsed) THEN
            CALL ERRMSS ('More than one CELSE found')
            WRITE (blabla, '("on line", i6)') iline
            CALL errmss (TRIM (blabla))
            STOP 0
          ELSE
            elsed = .TRUE.
          ENDIF
        ENDIF
        !IF (LEN_TRIM (card) > 5) THEN
        !  CALL ERRMSS ('Extra characters after CELSE')
        !  WRITE (blabla, '("on line", i6)') iline
        !  CALL errmss (TRIM (blabla))
        !  STOP 0
        !ENDIF
        IF (.NOT. inside) THEN
          CALL errmss ('CELSE found while not in conditional')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
        IF (inside2) THEN
! Level 2
          IF (.NOT. found2) THEN
            active2 = .TRUE.
            found2 = .TRUE.
          ELSE
            active2 = .FALSE.
          ENDIF
          CALL out (echo, card, first, star2, msg, active.AND.active2)
        ELSE
! Level 1
          IF (.NOT. found) THEN
            active = .TRUE.
            found = .TRUE.
          ELSE
```

```
            active = .FALSE.
          ENDIF
          CALL out (echo, card, first, star, msg, active)
        ENDIF
c
      ELSEIF (c7(1:7) == cendif) THEN
c
c CENDIF
c
        !IF (LEN_TRIM (card) > 6) THEN
        !  CALL ERRMSS ('Extra characters after CENDIF')
        !  WRITE (blabla, '("on line", i6)') iline
        !  CALL errmss (TRIM (blabla))
        !  STOP 0
        !ENDIF
        IF (inside2) THEN
! Level 2
          inside2 = .FALSE.
          nbr2 = 0
          CALL out (echo, card, first, star2, msg, active.AND..TRUE.)
          active2 = .FALSE.
          found2 = .FALSE.
          star2 = .FALSE.
          negative2 = .FALSE.
          elsed2 = .FALSE.
        ELSEIF (inside) THEN
! Level 1
          inside = .FALSE.
          nbr = 0
          CALL out (echo, card, first, star, msg, .TRUE.)
          active = .FALSE.
          found = .FALSE.
          star = .FALSE.
          negative = .FALSE.
          elsed = .FALSE.
        ELSE
          CALL errmss ('CENDIF found while not in conditional')
          WRITE (blabla, '("on line", i6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ENDIF
c
      ELSE
c
c Text line
c
        IF (inside2) THEN
          IF (active .AND. active2) THEN
            WRITE (6, '(A)') TRIM (card)
          ELSEIF (star2 .and. echo) THEN
            WRITE (6, '(A)') TRIM (first//card(2:))
          ENDIF
        ELSEIF (inside) THEN
          IF (active) THEN
            WRITE (6, '(A)') TRIM (card)
          ELSEIF (star .and. echo) THEN
            WRITE (6, '(A)') TRIM (first//card(2:))
          ENDIF
        ELSE
          WRITE (6, '(A)') TRIM (card)
        ENDIF
c
      ENDIF
c
      GO TO 1
c
c End of input file
c
  999 IF (inside) THEN
        CALL errmss ('EOF reached while within a CIF construct')
        STOP 0
      ENDIF
c
c Return 1 on normal execution (for use in PERL scripts).
c (When an error message is issued, the return code is 0).
c
      STOP 1
c
      END PROGRAM epx_filter
c=======================================================================
      SUBROUTINE errmss (msg)
c
      IMPLICIT NONE
c
      CHARACTER (LEN=*) :: msg
c
      WRITE (6,*) "*** ERROR *** "//msg
c
      END SUBROUTINE errmss
c=======================================================================
      SUBROUTINE usage
c
      IMPLICIT NONE
c
      WRITE (6,*) "Usage: epx_filter [-c<character>] mot1 [mot2 ...]"
      WRITE (6,*) "                  <input >output"
c
      END SUBROUTINE usage
c=======================================================================
      SUBROUTINE to_upper (s)
c
      IMPLICIT NONE
c
      CHARACTER (LEN=*), INTENT(INOUT) :: s
c
      INTEGER(4) :: i, k
c
      DO i = 1, LEN_TRIM (s)
        k = ICHAR (s (i:i))
        IF (k >= 97 .and. k <= 122) THEN
          s (i:i) = CHAR (k - 32)
        ENDIF
      END DO
c
      END SUBROUTINE to_upper
c=======================================================================
      SUBROUTINE get_passwords (card, npt, passt, iline, blabla)
c
      IMPLICIT NONE
c
      INTEGER(4), PARAMETER :: pwdlen = 16 ! Max. length of a password
      INTEGER(4), PARAMETER :: npasst = 9  ! Max. number of passwords
c                                          ! in a line of text file
c
      INTEGER(4), PARAMETER :: crdlen = 132! Max. length of a text 'card'
      CHARACTER (LEN=crdlen) :: c
c
      CHARACTER (LEN=*), INTENT(IN) :: card
      INTEGER(4), INTENT(OUT) :: npt
      CHARACTER (LEN=pwdlen),DIMENSION(npasst), INTENT(OUT) :: passt
      INTEGER(4), INTENT(IN) :: iline
      CHARACTER (LEN=*), INTENT(OUT) :: blabla
c
      INTEGER(4) :: i, k
c
      c = TRIM (card)
      npt = 0
c
      DO
        i = SCAN (TRIM (c), ' ')
        IF (i > 0) THEN
          npt = npt + 1
          IF (npt > npasst) THEN
            CALL errmss ('Too many passwords in text file')
            WRITE (blabla, '("on line", i6)') iline
            CALL errmss (TRIM (blabla))
            STOP 0
          ENDIF
          passt (npt) = c (1 : i-1)
          CALL to_upper (passt (npt))
          c = ADJUSTL (c (i+1 :))
        ELSE
          IF (LEN_TRIM (c) > 0) THEN
            npt = npt + 1
            IF (npt > npasst) THEN
              CALL errmss ('Too many passwords in text file')
              WRITE (blabla, '("on line", i6)') iline
              CALL errmss (TRIM (blabla))
              STOP 0
            ENDIF
            passt (npt) = TRIM (c)
            CALL to_upper (passt (npt))
          ENDIF
          EXIT
        ENDIF
      END DO
c
      END SUBROUTINE get_passwords
c=======================================================================
      SUBROUTINE compare_passwords (nbr, npt, passt, iline, blabla)
c
      IMPLICIT NONE
c
      INTEGER(4), PARAMETER :: pwdlen = 16 ! Max. length of a password
      INTEGER(4), PARAMETER :: npasst = 8  ! Max. number of passwords
c                                          ! in a line of text file
      INTEGER(4), PARAMETER :: nbran  = 10 ! Max. n. of branches in a CIF
c                                          ! construct
c
      INTEGER(4), INTENT(IN) :: nbr
      INTEGER(4), DIMENSION(nbran), INTENT(IN) :: npt
      CHARACTER (LEN=pwdlen),
     &           DIMENSION(npasst, nbran) :: passt  ! Passwords in each
c                                                    branch of txtfile
      INTEGER(4), INTENT(IN) :: iline
      CHARACTER (LEN=*), INTENT(OUT) :: blabla
c
      CHARACTER (LEN=pwdlen) :: passj, passl
c
      INTEGER(4) :: i, j, k, l
c
      DO i = 2, nbr
        DO j = 1, npt (i)
          passj = passt (j, i)
          DO k = 1, nbr - 1
            DO l = 1, npt (k)
              passl = passt (l, k)
              IF (l /= j .OR. k /= i) THEN
                IF (passj == passl) THEN
                  CALL errmss ('Repeated password ')
                  WRITE (blabla, '(a, "on line", I6)') passj, iline
                  CALL errmss (TRIM (blabla))
                  STOP 0
                ENDIF
              ENDIF
            END DO
          END DO
        END DO
      END DO
c
      END SUBROUTINE compare_passwords
c=======================================================================
      SUBROUTINE out (echo, card, first, star, msg, active)
c
      IMPLICIT NONE
c
      INTEGER(4), PARAMETER :: crdlen = 132! Max. length of a text 'card'
c
      LOGICAL(4), INTENT(IN) :: echo
      CHARACTER (LEN=crdlen), INTENT(IN) :: card
      CHARACTER (LEN=1), INTENT(IN) :: first
      LOGICAL(4), INTENT(IN) :: star
      CHARACTER (LEN=21), INTENT(IN) :: msg
      LOGICAL(4), INTENT(IN) :: active
c
      IF (star) THEN
        IF (echo) WRITE (6, '(A)') TRIM (first//card(2:))
      ELSE
        IF (echo) THEN
          WRITE (6, '(A)') TRIM (first//card(2:))
          IF (.NOT. active) THEN
            WRITE (6, '(A)') first//msg
          ENDIF
        ENDIF
      ENDIF
c
```

```
      END SUBROUTINE out
c=======================================================================
      SUBROUTINE verify_passwords (npa, passw, npt, passt, active,
     &                             iline, blabla)
c
      IMPLICIT NONE
c
      INTEGER(4), PARAMETER :: pwdlen = 16 ! Max. length of a password
      INTEGER(4), PARAMETER :: npassw = 8  ! Max. number of passwords
c                                          ! given in command line
      INTEGER(4), PARAMETER :: npasst = 8  ! Max. number of passwords
c                                          ! in each line of text file
c
      INTEGER(4), INTENT(IN) :: npa
      CHARACTER (LEN=pwdlen), DIMENSION(npassw), INTENT(IN) :: passw
      INTEGER(4), INTENT(IN) :: npt
      CHARACTER (LEN=pwdlen), DIMENSION(npasst), INTENT(IN) :: passt
      LOGICAL(4), INTENT(INOUT) :: active
      INTEGER(4), INTENT(IN) :: iline
      CHARACTER (LEN=80), INTENT(OUT) :: blabla
c
      INTEGER(4) :: i, j
      LOGICAL(4) :: match
      CHARACTER (LEN=pwdlen) :: passi, passj
c
c Compare current text line passwords with cmdline passwords,
c to see if there is at least one match.
c
      match = .FALSE.
      DO i = 1, npt
        passi = passt (i)
        DO j = 1, npa
          passj = passw (j)
          IF (passj == passi) THEN
            match = .TRUE.
            GO TO 1
          ENDIF
        END DO
      END DO

c
    1 IF (.NOT. active) THEN
c
c If at least one match was found, set active = .TRUE.
c
        IF (match) THEN
          active = .TRUE.
        ENDIF
      ELSE
c
c If a match was found, give error message,
c otherwise set active = .FALSE.
c
        IF (match) THEN
          CALL errmss ('Specified passwords match '//
     &                 'more than one branch')
          WRITE (blabla, '("on line ",I6)') iline
          CALL errmss (TRIM (blabla))
          STOP 0
        ELSE
          active = .FALSE.
        ENDIF
      ENDIF
c
      END SUBROUTINE verify_passwords
```

epx_ftp_getfiles.pl

```
#-------------------------------------------------------------------------------
# Get EUROPLEXUS tar file with latest evolutions from central mirror site
# and split it to local directory, ready to start evolution.
#-------------------------------------------------------------------------------
# Modules needed
#
use HTTP::Date;
use NET::FTP;
#-------------------------------------------------------------------------------
# Redirect STDOUT and STDERR to evol.log (Fixed path!!!!)
#
$logfile = "C:\\EUROPLEXUS\\Fromcentral\\evol_getfiles.log";
open STDOUT, ">>$logfile";
open STDERR, ">>$logfile";
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  die "The EUROPLEXUS environment variable is undefined!\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  die "The EUROPLEXUS directory: $epx does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Check existence of evolution version file
#
$evonumfile = "$epx\\VERSION.txt";         # Evolution version file
if (! -r $evonumfile) {
  die "The evolution number file $evonumfile does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Read the number of the previous evolution
#
$evonum = `cat $evonumfile`;
chop $evonum;
print "\n===> The current evolution index is  : \# $evonum\n";
$numok = $evonum + 1;
print "===> The next evolution index must be: \# $numok\n";
#-------------------------------------------------------------------------------
# Get and print current date and time
#
$time = time;                          # Machine time (seconds since epoch)
$gmttime = time2str($time);            # Universal (GMT) date and time
$loctime = time2iso($time);            # This requires function time2iso to be
                                       # exported in module HTTP\Date.pm (is not
                                       # exported by default!
print "\n\n";
print "=======================================";
print "=======================================\n";
print "===> EUROPLEXUS copying EUROPLEXUS evolution file on: $gmttime\n";
print "===> Local date/time: $loctime\n";
#-------------------------------------------------------------------------------
# Initialize times
#
($user0, $system0, $cuser0, $csystem0) = times;
#-------------------------------------------------------------------------------
# Default values
#
#$ftpdir = "C:\\IIS\\FTP\\EUROPLEXUS"; # Directory receiving FTP files
$ftpdir = "$epx\\FTP"; # Directory receiving FTP files
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 0
#
if ($#ARGV != -1) {                    # Index of last argument must be = -1
  die "Usage: epx_ftp_getfiles\007\n";
}
#-------------------------------------------------------------------------------
# Make sure we are in the directory that will receive FTP files
#
$curdir = `pwd`;
chop $curdir;
if ( $curdir ne $ftpdir ) {
  print "\nChanging directory to $ftpdir.\n";
  chdir "$ftpdir" ||
  die "Can't chdir to $ftpdir!\007\n";
}
#-------------------------------------------------------------------------------
# Access central mirror site, and copy the evolution file to $ftpdir
#
$ftp = Net::FTP->new("ftp.cea.fr", Debug => 0);
$ftp->login("europlex","euro35!");
$ftp->cwd("evol");
$ftp->binary;
$file = "europlexus.tar";
print "Copying file $file ...\n";
$ftp->get("$file");
$ftp->quit;
#-------------------------------------------------------------------------------
# Extract evolution files from tar archive and remove archive
#
print "Untarring file $file ...\n";
system("tar xvf $file");
print "Removing file $file ...\n";
unlink $file;
#-------------------------------------------------------------------------------
# Remove any obsolete evolution files:
#
# Treat all the evolution "resume" files, i.e., files of the
# form R_nnnnxjjmmmyy.txt in $ftpdir. These contain the list of
# evolution packages relative to each evolution, i.e names of files of the
# form [ISBMV]_NNNNxDDMMMYY.tar.gz that must also be in $ftpdir.
#
opendir(DIR, $ftpdir) || die "Can't open $ftpdir\n";
@FileNames = sort readdir(DIR);
closedir(DIR);
#
foreach $file (@FileNames) {
  $_ = $file;
  $dirf = "$ftpdir\\$file";
  if ( -r $dirf ) {
#-------------------------------------------------------------------------------
    # NOTE: the preceding test on file existence is necessary, because
    # treatment of a file in the list involves treatment (and deletion)
    # of all its matching files, if any!
    #
    if ( m/^R_\d\d\d\dx\d\d[A-Z][a-z][a-z]\d\d.txt$/ ) {
      print "\n===> Evolution resume file found: $file\n";
#-------------------------------------------------------------------------------
      # Evolution resume file found: open it and read names of
      # the associated evolution packages (I_, S_, B_, M_, V_), of which
      # zero to five must be listed, and which must be present in the
      # FTP directory.
      #
      # The case zero corresponds to an evolution which is empty because
      # it failed at the central mirror site. In this case the trace is
      # sent anyway (it will contain the error message). The only effects
      # on the mirror site are in this case:
      #   - the evolution counter is incremented;
      #   - the resume (R_) and trace (T_) files are moved to $trcdir
      #
      $rfile = $file;
      $rbase = $rfile; $rbase =~ s/\.txt$//;
      #
      $numdate = $rbase; $numdate =~ s/^R_//;     # Number & date (NNNNxDDMMMYY)
      $date = $numdate; $date =~ s/^.....//;      # Date (DDMMMYY)
      $num = $numdate; $num =~ s/x.......$//;     # Number (with leading 0s)
      $lognum = $num;
      $num =~ s/^0*//;                            # Number (no leading 0s)
      print "The evolution index is: $num\n";
      #
#-------------------------------------------------------------------------------
      # Is the index of this evolution obsolete?
      #
      if ($num < $numok) {
        print "Evolution number $num is obsolete and is being removed ...\n";
        #
        $dirr = "$ftpdir\\$rfile";
        #
        open (RESUME, $dirr);
        while (<RESUME>) {
          chop;
          if ( m/^[ISBMV]_\d\d\d\dx\d\d[A-Z][a-z][a-z]\d\d.tar.gz$/ ) {
            $entry = $_;
            $entry =~ s/^._...../;
            $entry =~ s/\.tar\.gz$//;
            if ($entry eq $date) {
              #
```

```
            # The entry in the resume file matches the evolution date
            #
            if    ( m/^I/ ) {$ifile = $_}
            elsif ( m/^S/ ) {$sfile = $_}
            elsif ( m/^B/ ) {$bfile = $_}
            elsif ( m/^M/ ) {$mfile = $_}
            elsif ( m/^V/ ) {$vfile = $_}
            else {
              $errmsg =
              "The resume file contains an ambiguous name: $_!\007\n";
              print "$errmsg";
              die "$errmsg";
            }
          }
        }
      }
      close RESUME;
#-------------------------------------------------------------------------------
      # Verify that the trace file corresponding to the resume file
      # is present in $ftpdir
      #
      $tfile = $rfile; $tfile =~ s/^R/T/;
      $dirt = "$ftpdir\\$tfile";
      #
      if (! -r $dirt) {
        $errmsg = "The trace file $tfile missing in $ftpdir!\007\n";
        print "$errmsg";
        die "$errmsg";
      }
#-------------------------------------------------------------------------------
      # Remove the existing packages (I_, S_, B_, M_, V_), if any,
      # from the directory $ftpdir.
      #
      if (defined($ifile)) {
        $diri = "$ftpdir\\$ifile";
        if ( -r $diri ) {
          unlink $diri;
        }
        else {
          $errmsg = "File $ifile listed in $rfile missing in $ftpdir!\007\n";
          print "$errmsg";
          die "$errmsg";
        }
      }
      if (defined($sfile)) {
        $dirs = "$ftpdir\\$sfile";
        if ( -r $dirs ) {
          unlink $dirs;
        }
        else {
          $errmsg = "File $sfile listed in $rfile missing in $ftpdir!\007\n";
          print "$errmsg";
          die "$errmsg";
        }
      }
      if (defined($bfile)) {
        $dirb = "$ftpdir\\$bfile";
        if ( -r $dirb ) {
          unlink $dirb;
        }
        else {
          $errmsg = "File $bfile listed in $rfile missing in $ftpdir!\007\n";
          print "$errmsg";
          die "$errmsg";
        }
      }
      if (defined($mfile)) {
        $dirm = "$ftpdir\\$mfile";
        if ( -r $dirm ) {
          unlink $dirm;
        }
        else {
          $errmsg = "File $mfile listed in $rfile missing in $ftpdir!\007\n";
          print "$errmsg";
          die "$errmsg";
        }
      }
      if (defined($vfile)) {
        $dirv = "$ftpdir\\$vfile";
        if ( -r $dirv ) {
          unlink $dirv;
        }
        else {
          $errmsg = "File $vfile listed in $rfile missing in $ftpdir!\007\n";
          print "$errmsg";
          die "$errmsg";
        }
      }
#-------------------------------------------------------------------------------
      # Remove the resume file and trace file from the directory $ftpdir.
      #
      unlink $dirr;
      unlink $dirt;
#-------------------------------------------------------------------------------
      # Reset the variables that are tested through "defined",
      # in case another evolution has to be performed
      #
      undef $ifile; undef $sfile; undef $bfile; undef $mfile; undef $vfile;
    }
    else {
      print "Evolution number $num is new and is kept ...\n";
    }
  }
 }
}
#-------------------------------------------------------------------------------
# Get and print current date and time
#
$time = time;                       # Machine time (seconds since epoch)
$gmttime = time2str($time);         # Universal (GMT) date and time
$loctime = time2iso($time);         # This requires function time2iso to be
                                    # exported in module HTTP\Date.pm (is not
                                    # exported by default!
print "===> EUROPLEXUS ended copying evolution file on: $gmttime\n";
print "===> Local date/time: $loctime\n";
print "========================================";
print "========================================\n";
#-------------------------------------------------------------------------------
# Clean up and exit
#
close STDOUT;
close STDERR;
#
exit;
```

epx_ftp_putexe.pl

```
#-------------------------------------------------------------------------------
# Update executable module on the machine outside the firewall
#-------------------------------------------------------------------------------
# Modules needed
#
use HTTP::Date;
use NET::FTP;
#-------------------------------------------------------------------------------
# Redirect STDOUT and STDERR to evol.log (Fixed path!!!!)
#
$logfile = "C:\\EUROPLEXUS\\Fromcentral\\evol_putexe.log";
open STDOUT, ">>$logfile";
open STDERR, ">>$logfile";
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Get and print current date and time
#
$time = time;                       # Machine time (seconds since epoch)
$gmttime = time2str($time);         # Universal (GMT) date and time
$loctime = time2iso($time);         # This requires function time2iso to be
                                    # exported in module HTTP\Date.pm (is not
                                    # exported by default!
print "\n";
print "========================================";
print "========================================\n";
print "===> EUROPLEXUS updating online executable on: $gmttime\n";
print "===> Local date/time: $loctime\n";
#-------------------------------------------------------------------------------
# Initialize times
#
($user0, $system0, $cuser0, $csystem0) = times;
#-------------------------------------------------------------------------------
# Default values
#
$exefile = "europlexus.exe.gz";
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 0 or 1
#
if ($#ARGV >= 1) {                    # Index of last argument must be = -1 or 0
  die "Usage: epx_ftp_putexe [-w] [-M] [-c]\007\n";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#       -w      copy QuickWin executable instead of Console
#       -M      copy MPI executable instead of Console
#       -c      copy -check executable instead of Console
#
while ($ARGV[0] =~ /^-/) {
  $_ = shift;
  if (/^-w(.*)/) {
    $exefile = "europlexusqw.exe.gz";
    printf "Command line switch: QuickWin\n";
  }
  elsif (/^-M(.*)/) {
    $exefile = "europlexus_mpi.exe.gz";
    printf "Command line switch: MPI\n";
  }
  elsif (/^-c(.*)/) {
    $exefile = "europlexus_check.exe.gz";
    printf "Command line switch: -check\n";
  }
  else {
    $errmsg = "ERROR: unknown switch: $_\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  die "The EUROPLEXUS environment variable is undefined!\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  die "The EUROPLEXUS directory: $epx does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Set and check existence of auxiliary directories
#
$evodir = "$epx\\Fromcentral";              # Directory used for evolution
if ( ! -d $evodir) {
  die "The EUROPLEXUS directory: $evodir does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Make sure we are in the directory that will sending FTP files
#
$curdir = `pwd`;
chop $curdir;
if ( $curdir ne $evodir ) {
  print "\nChanging directory to $evodir.\n";
  chdir "$evodir" ||
  die "Can't chdir to $evodir!\007\n";
}
#-------------------------------------------------------------------------------
# Set and check existence of file to be copied
#
if ( ! -r $exefile) {
  die "The EUROPLEXUS executable: $exefile does not exist!\007\n";
}
#-------------------------------------------------------------------------------
```

```
# Access machine outside the firewall, and copy executable
#
#$ftp = Net::FTP->new("europlexus.jrc.it", Debug => 0);
#$ftp->login("europlexus","la,mi!");
#$ftp->cwd("/srv/sites/plexus/web_site/consortium/exe");
#$ftp->binary;
print "Updating file $exefile ...\n";
#system("pscp -q -l europlexus -pw la,mi! -batch $exefile europlexus\@elsa2.jrc.
it:/srv/sites/plexus/web_site/consortium/exe");
#system("pscp -q -l europlexus -pw la,mi! -batch $exefile europlexus\@europlexus
.jrc.ec.europa.eu:/srv/sites/plexus/web_site/consortium/exe");
system("pscp -q -l europlexus -pw la,mi! -batch $exefile europlexus\@139.191.1.4
7:/srv/sites/plexus/web_site/consortium/exe");
#$ftp->delete("$exefile");
#$ftp->put("$exefile");
#$ftp->quit;
#-------------------------------------------------------------------------------
# Get and print current date and time
#
$time = time;                       # Machine time (seconds since epoch)
$gmttime = time2str($time);         # Universal (GMT) date and time
$loctime = time2iso($time);         # This requires function time2iso to be
                                    # exported in module HTTP\Date.pm (is not
                                    # exported by default!
print "===> EUROPLEXUS ended updating online executable file on: $gmttime\n";
print "===> Local date/time: $loctime\n";
print "========================================";
print "========================================\n";
#-------------------------------------------------------------------------------
# Clean up and exit
#
close STDOUT;
close STDERR;
#
exit;
```

epx_ftp_putexe_64.pl

```
#-------------------------------------------------------------------------------
# Update 64-bit executable module on the machine outside the firewall
#-------------------------------------------------------------------------------
# Modules needed
#
use HTTP::Date;
use NET::FTP;
#-------------------------------------------------------------------------------
# Redirect STDOUT and STDERR to evol.log (Fixed path!!!!)
#
$logfile = "E:\\EUROPLEXUS\\Fromcentral\\evol_putexe.log";
open STDOUT, ">>$logfile";
open STDERR, ">>$logfile";
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Get and print current date and time
#
$time = time;                       # Machine time (seconds since epoch)
$gmttime = time2str($time);         # Universal (GMT) date and time
$loctime = time2iso($time);         # This requires function time2iso to be
                                    # exported in module HTTP\Date.pm (is not
                                    # exported by default!
print "\n";
print "========================================";
print "========================================\n";
print "===> EUROPLEXUS updating online 64-bit executable on: $gmttime\n";
print "===> Local date/time: $loctime\n";
#-------------------------------------------------------------------------------
# Initialize times
#
($user0, $system0, $cuser0, $csystem0) = times;
#-------------------------------------------------------------------------------
# Default values
#
$exefile = "europlexus64.exe.gz";
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 0 or 1
#
if ($#ARGV >= 1) {                    # Index of last argument must be = -1 or 0
  die "Usage: epx_ftp_putexe_64 [-M]\007\n";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#       -M      copy 64-bit MPI executable instead of Console
#
while ($ARGV[0] =~ /^-/) {
  $_ = shift;
  if (/^-M(.*)/) {
    $exefile = "europlexus64_mpi.exe.gz";
    printf "Command line switch: MPI\n";
  }
  else {
    $errmsg = "ERROR: unknown switch: $_\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  die "The EUROPLEXUS environment variable is undefined!\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  die "The EUROPLEXUS directory: $epx does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Set and check existence of auxiliary directories
#
$evodir = "$epx\\Fromcentral";             # Directory used for evolution
if ( ! -d $evodir) {
  die "The EUROPLEXUS directory: $evodir does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Make sure we are in the directory that will sending FTP files
#
$curdir = `pwd`;
chop $curdir;
if ( $curdir ne $evodir ) {
  print "\nChanging directory to $evodir.\n";
  chdir "$evodir" ||
  die "Can't chdir to $evodir!\007\n";
}
#-------------------------------------------------------------------------------
# Set and check existence of file to be copied
#
if ( ! -r $exefile) {
  die "The EUROPLEXUS executable: $exefile does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Access machine outside the firewall, and copy executable
#
#$ftp = Net::FTP->new("europlexus.jrc.it", Debug => 0);
#$ftp->login("europlexus","la,mi!");
#$ftp->cwd("/srv/sites/plexus/web_site/consortium/exe");
#$ftp->binary;
print "Updating file $exefile ...\n";
#system("pscp -q -l europlexus -pw la,mi! -batch $exefile europlexus\@elsa2.jrc.
it:/srv/sites/plexus/web_site/consortium/exe");
#system("pscp -q -l europlexus -pw la,mi! -batch $exefile europlexus\@europlexus
.jrc.ec.europa.eu:/srv/sites/plexus/web_site/consortium/exe");
system("pscp -q -l europlexus -pw la,mi! -batch $exefile europlexus\@139.191.1.4
7:/srv/sites/plexus/web_site/consortium/exe");
#$ftp->delete("$exefile");
#$ftp->put("$exefile");
#$ftp->quit;
#-------------------------------------------------------------------------------
# Get and print current date and time
#
$time = time;                       # Machine time (seconds since epoch)
$gmttime = time2str($time);         # Universal (GMT) date and time
$loctime = time2iso($time);         # This requires function time2iso to be
                                    # exported in module HTTP\Date.pm (is not
                                    # exported by default!
print "===> EUROPLEXUS ended updating online executable file on: $gmttime\n";
print "===> Local date/time: $loctime\n";
print "========================================";
print "========================================\n";
#-------------------------------------------------------------------------------
# Clean up and exit
#
close STDOUT;
close STDERR;
#
exit;
```

epx_ftp_putman.pl

```
#-------------------------------------------------------------------------------
# Update manuals (html and pdf) on the machine outside the firewall
#-------------------------------------------------------------------------------
# Modules needed
#
use HTTP::Date;
use NET::FTP;
#-------------------------------------------------------------------------------
# Redirect STDOUT and STDERR to evol.log (Fixed path!!!!)
#
$logfile = "C:\\EUROPLEXUS\\Fromcentral\\evol_putman.log";
open STDOUT, ">>$logfile";
open STDERR, ">>$logfile";
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Get and print current date and time
#
$time = time;                       # Machine time (seconds since epoch)
$gmttime = time2str($time);         # Universal (GMT) date and time
$loctime = time2iso($time);         # This requires function time2iso to be
                                    # exported in module HTTP\Date.pm (is not
                                    # exported by default!
print "\n";
print "========================================";
print "========================================\n";
print "===> EUROPLEXUS updating online manuals on: $gmttime\n";
print "===> Local date/time: $loctime\n";
#-------------------------------------------------------------------------------
# Initialize times
#
($user0, $system0, $cuser0, $csystem0) = times;
#-------------------------------------------------------------------------------
# Default values
#
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 0
#
if ($#ARGV != -1) {                       # Index of last argument must be = -1
  die "Usage: epx_ftp_putman\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  die "The EUROPLEXUS environment variable is undefined!\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  die "The EUROPLEXUS directory: $epx does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Set and check existence of auxiliary directories
```

```
#
$evodir = "$epx\\Fromcentral";             # Directory used for evolution
if ( ! -d $evodir) {
  die "The EUROPLEXUS directory: $evodir does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Make sure we are in the directory that will sending FTP files
#
$curdir = `pwd`;
chop $curdir;
if ( $curdir ne $evodir ) {
  print "\nChanging directory to $evodir.\n";
  chdir "$evodir" ||
  die "Can't chdir to $evodir!\007\n";
}
#-------------------------------------------------------------------------------
# Set and check existence of files to be copied
#
$pdffile = "manual.pdf";
if ( ! -r $pdffile) {
  die "The EUROPLEXUS manual (pdf): $pdffile does not exist!\007\n";
}
$htmldir = "hacha";
if ( ! -d $htmldir) {
  die "The EUROPLEXUS manual (html) dir: $htmldir does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Build up list of files in $htmldir
#
opendir (HACHA, "$htmldir");
@files = readdir (HACHA);
closedir (HACHA);
#-------------------------------------------------------------------------------
# Access machine outside the firewall, and copy manuals
#
#$ftp = Net::FTP->new("europlexus.jrc.it", Debug => 0);
#$ftp->login("europlexus","la,mi!");
#
# pdf version
#
#$ftp->cwd("/srv/sites/plexus/web_site/consortium/manual_pdf");
#$ftp->binary;
print "Updating file $pdffile ...\n";
#system("pscp -q -l europlexus -pw la,mi! -batch $pdffile europlexus\@elsa2.jrc.
it:/srv/sites/plexus/web_site/consortium/manual_pdf");
#system("pscp -q -l europlexus -pw la,mi! -batch $pdffile europlexus\@europlexus
.jrc.ec.europa.eu:/srv/sites/plexus/web_site/consortium/manual_pdf");
system("pscp -q -l europlexus -pw la,mi! -batch $pdffile europlexus\@139.191.1.4
7:/srv/sites/plexus/web_site/consortium/manual_pdf");
#$ftp->delete("$pdffile");
#$ftp->put("$pdffile");
#
# html version (hacha)
#
#$ftp->cwd("/srv/sites/plexus/web_site/public/manual_html");
print "Updating all files in $htmldir:\n";
#system("pscp -q -l europlexus -pw la,mi! -batch $htmldir\\*.* europlexus\@elsa2
.jrc.it:/srv/sites/plexus/web_site/public/manual_html");
#system("pscp -q -l europlexus -pw la,mi! -batch $htmldir\\*.* europlexus\@europ
lexus.jrc.ec.europa.eu:/srv/sites/plexus/web_site/public/manual_html");
system("pscp -q -l europlexus -pw la,mi! -batch $htmldir\\*.* europlexus\@139.19
1.1.47:/srv/sites/plexus/web_site/public/manual_html");
#
#foreach $file (@files) {
#  $_ = $file;
#  if ( m/^\./ ) {
#    # Do not copy . and .. files !
#  }
#  else {
#    print "  File $file ...\n";
#    $fromfile = "$htmldir\\$file";
#    $ftp->delete("$file");
#    $ftp->put("$fromfile", "$file");
#  }
#}
#$ftp->quit;
#-------------------------------------------------------------------------------
# Get and print current date and time
#
$time = time;                       # Machine time (seconds since epoch)
$gmttime = time2str($time);         # Universal (GMT) date and time
$loctime = time2iso($time);         # This requires function time2iso to be
                                    # exported in module HTTP\Date.pm (is not
                                    # exported by default!
print "===> EUROPLEXUS ended updating online manual files on: $gmttime\n";
print "===> Local date/time: $loctime\n";
print "========================================";
print "========================================\n";
#-------------------------------------------------------------------------------
# Clean up and exit
#
close STDOUT;
close STDERR;
#
exit;
```

epx_get.pl

```
#-------------------------------------------------------------------------------
# Retrieve Fortran source|include file from EUROPLEXUS library
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$ext = "ff";                            # By default, source file
$dir = "source";                        # By default, source directory
$copy = "cp -p";                        # Alternative: use copy
#-------------------------------------------------------------------------------
# Check number of arguments, must be 1 or 2 or 3
#
if ($#ARGV < 0 || $#ARGV > 2) {       # Index of last argument must be 0, 1 or 2
  die "Usage: epx_get [-q] [-i] [-o] [-b] [-m] [-v] name[.<extension>]\007\n";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#       -q     quiet (do not print subsequent switch echo and file copy
#              confirmation)
#       -i     include file, i.e. ".inc" file, not source (.ff) file
#       -o     module object, i.e. ".mod" file, not source (.ff) file
#       -b     benchmark, i.e. ".epx" file, not source (.ff) file
#       -m     manual, i.e. ".ttx" file, not source (.ff) file
#       -u     utility file, i.e. ".pl" file, not source (.ff) file
#       -v     validation file, i.e. ".vld" file, not source (.ff) file
#
while ($ARGV[0] =~ /^-/) {
  $_ = shift;
  if (/^-q(.*)/) {
    $quiet = "yes";
  }
  elsif (/^-i(.*)/) {
    $ext = "inc";
    $dir = "include";
    if (! defined($quiet)) {printf "Command line switch: include file\n";}
  }
  elsif (/^-o(.*)/) {
    $ext = "mod";
    $dir = "module";
    if (! defined($quiet)) {printf "Command line switch: module file\n";}
  }
  elsif (/^-b(.*)/) {
    $ext = "epx";
    $ext2 = "msh";
    $ext3 = "zip";
    $dir = "bench";
    if (! defined($quiet)) {printf "Command line switch: benchmark file\n";}
  }
  elsif (/^-m(.*)/) {
    $ext = "ttx";
    $dir = "manual";
    if (! defined($quiet)) {printf "Command line switch: manual file\n";}
  }
  elsif (/^-u(.*)/) {
    $ext = "pl";
    $dir = "util";
    if (! defined($quiet)) {printf "Command line switch: utility (Perl) file\n";
}
  }
  elsif (/^-v(.*)/) {
    $ext = "vld";
    $ext3 = "zip";
    $dir = "validate";
    if (! defined($quiet)) {printf "Command line switch: validation file\n";}
  }
  else {
    die "ERROR: unknown switch: $_\n";
  }
}
#-------------------------------------------------------------------------------
# Check if the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  die "The EUROPLEXUS environment variable is undefined!\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  die "The EUROPLEXUS directory: $epx does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Set up file name
#
$name = $ARGV[0];                           # Get fist and only argument
$base = $name;
$base =~ s/\.$ext$//;                       # Remove extension if present
$file = "$base.$ext";
#-------------------------------------------------------------------------------
# Verify that the file does not exist already
#
if ( -r $file ) {                           # Test if target exists already
  die "File $file exists already in current directory!\007\n";
}
if (defined($file2)) {
  if ( -r $file2 ) {                        # Test if target 2 exists already
    die "File $file2 exists already in current directory!\007\n";
  }
}
#-------------------------------------------------------------------------------
# Verify that the relevant file exists in the EUROPLEXUS directory
#
$from = "$epx\\$dir\\$file";                # Full file name
if ( ! -r $from ) {
  die "File $file does not exist in $epx\\$dir!\007\n";
}
if (defined($ext2)) {
  $file2 = "$base.$ext2";
  $from2 = "$epx\\$dir\\$file2";            # Full file 2 name
  #
  # Mesh file may or may not exist in benchmarks directory
  #
  if ( -r $from2 ) {
    if ( -r $file2 ) {                      # Test if target 2 exists already
      die "File $file2 exists already in current directory!\007\n";
    }
  }
}
if (defined($ext3)) {
  $file3 = "$base.$ext3";
  $from3 = "$epx\\$dir\\$file3";            # Full file 3 name
  #
  # Zip file may or may not exist in benchmarks or validations directory
  #
  if ( -r $from3 ) {
    if ( -r $file3 ) {                      # Test if target 3 exists already
      die "File $file3 exists already in current directory!\007\n";
    }
  }
}
#-------------------------------------------------------------------------------
```

```
# Copy file from the EUROPLEXUS to the current directory
#
system("$copy $from .");
if (! defined($quiet)) {print "File $from was copied to \".\".\n";}
if (defined($ext2)) {
  if ( -r $from2 ) {
    system("$copy $from2 .");
    if (! defined($quiet)) {print "File $from2 was copied to \".\".\n";}
  }
}
if (defined($ext3)) {
  if ( -r $from3 ) {
    system("$copy $from3 .");
    if (! defined($quiet)) {print "File $from3 was copied to \".\".\n";}
  }
}
#
exit;
```

epx_get_includers.pl

```
#-------------------------------------------------------------------------------
# Retrieve EUROPLEXUS includer file(s) from the sources directory
#-------------------------------------------------------------------------------
use File::Basename;                     # Gives access to fileparse function
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$rout = "_includers.txt";
$temp = "_temp_.txt";
$copy = "cp -p";
$del  = "rm -f";
$atts = 0;
$errfil = "epx_get_includers.err";
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 0
#
if ($#ARGV != -1) {                     # Index of last argument must be = -1
  $errmsg = "Usage: epx_get_includers\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
$gnudir = $ENV{'GNUDIR'};
if ( ! defined $gnudir) {
  $errmsg = "The GNUDIR environment variable is undefined!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
$sort = "$gnudir\\sort.exe"; # Full path needed! Else uses MS-DOS's sort!
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  $errmsg = "The EUROPLEXUS environment variable is undefined!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  $errmsg = "The EUROPLEXUS directory: $epx does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Build array of include file names on current directory
#
@FileNames = glob ("*.inc");
#-------------------------------------------------------------------------------
# Build list of includers (source files that include any of the include files)
#
system ("$del $rout");
#
print "Building the includers list ...\n";
foreach $file (@FileNames) {
  print "  Searching includers of $file\n";
  system ("epx_grep -l \"$file\" >>$rout");
}
#-------------------------------------------------------------------------------
# Sort list of includers by eliminating multiple name occurrences
#
system ("$sort -u $rout >$temp");
system ("$del $rout");
#system ("ren $temp $rout");
system ("$copy $temp $rout");
system ("$del $temp");
$num = `cat $rout | wc -l`;
chop $num;
$num =~ s/ //g;
print "... includers list built and sorted ($num files)\n";
#-------------------------------------------------------------------------------
# Loop on routine(s) to be retrieved
#
$pref = "$epx\\source";
open (ROUTS, $rout);
while (<ROUTS>) {
  chop;
  $file = $_;
  #-----------------------------------------------------------------------------
  # Set up file name
  #
  ($name,$path,$suffix) = fileparse($file,"\.ff");
  $filnam = "$name$suffix";
  #-----------------------------------------------------------------------------
  if ( -r $filnam ) {
    #---------------------------------------------------------------------------
    # Routine file is already on current dir. Error message
    #
    print "ATTENTION! $filnam not retrieved since already in current dir!\n";
    $errmsg = "$filnam already present in current directory!\007\n";
#   &ERRFIL ($errfil, $errmsg);
    $atts = $atts + 1;
  }
  else {
    #---------------------------------------------------------------------------
    # Routine file is not on current dir. Retrieve it
    #
    print "Retrieving $filnam\n";
    system ("epx_get -q $name");
  }
}
close ROUTS;
#
if ($atts > 0) {
  print "Includers retrieval has successfully terminated.\n";
  print "(some includer files were already present in evolution dir).\n";
}
else {
  print "Includers retrieval has successfully terminated.\n";
}
#
exit;


#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_get_users.pl

```
# Get all users (direct and indirect) of a certain module
# Users must have been previously sought by the epx_modtree command
# Their list must be in file _users.txt on current directory.
#
# Files already on the current directory are NOT rewritten!
#
#-------------------------------------------------------------------------------
use File::Basename;                     # Gives access to fileparse function
#-------------------------------------------------------------------------------
$users = "_users.txt";
$tmp = "_tmp.txt";
open (INPUT, $users);
while (<INPUT>) {
  chop;
  $file = $_;
  ($base,$path,$suffix) = fileparse($file,"\.ff");
  $base = "$base\.ff";
  if ( ! -r $base ) { # Do not get file if already here
    $base =~ s/\.ff$//;
    system ("epx_get $base");
  }
  else {
    print "  ==> File $base not retrieved because it is already here !!!\007\n";
  }
}
close INPUT;
```

epx_grep.pl

```
#-------------------------------------------------------------------------------
# Search pattern in all Fortran source|include files from EUROPLEXUS library
# If the pattern contains spaces, enclose in in ""
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$ext = "ff";                            # By default, source files
$dir = "source";                        # By default, source directory
$opt = "-n";                            # By default, list line numbers
$arg = "";                              # Arguments for get - feault:no argumen
ts
$get = "0";                             # Flag wheather the files should also b
e copied
#-------------------------------------------------------------------------------
# Check number of arguments, must be 1 or 2 or 3
#
if ($#ARGV < 0 || $#ARGV > 2) {     # Index of last argument must be 0 or 1 or 2
  die "Usage: epx_grep [-i] [-l] [-b] [-m] [-u] [-h] [-v] [-g] [-?] [\"]pattern[
\"]\007\n";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#       -i     include file, i.e. ".inc" file, not source (.ff) file
#       -l     list file name only, not file name + found line
#       -b     benchmark, i.e. ".epx" file, not source (.ff) file
#       -m     manual, i.e. ".ttx" file, not source (.ff) file
#       -u     utility file, i.e. ".pl" file, not source (.ff) file
#       -h     history file, i.e. ".his" file, not source (.ff) file
#       -v     validation file, i.e. ".vld" file, not source (.ff) file
#       -g     get also the concerned files
#
while ($ARGV[0] =~ /^-/) {
  $_ = shift;
  if (/^-\?(.*)/) {
     printf "Search pattern in all Fortran source|include files from EUROPLEXUS
library\n";
     printf "If the pattern contains spaces, enclose in in \"\"\n";
     printf "Process optional switches:\n";
     printf " -i   include file, i.e. \".inc\" file, not source (.ff) file\n";
     printf " -l   list file name only, not file name + found line\n";
     printf " -b   benchmark, i.e. \".epx\" file, not source (.ff) file\n";
     printf " -m   manual, i.e. \".ttx\" file, not source (.ff) file\n";
     printf " -u   utility file, i.e. \".pl\" file, not source (.ff) file\n";
     printf " -h   history file, i.e. \".his\" file, not source (.ff) file\n";
     printf " -v   validation file, i.e. \".vld\" file, not source (.ff) file\n"
;
```

```
    printf " -g   get also the concerned files\n";
    exit;
  }
  elsif (/^-i(.*)/) {
    $ext = "inc";
    $dir = "include";
    $arg = "-i ";
#   printf "Command line switch: include files\n";
  }
  elsif (/^-l(.*)/) {
    $opt = "-l";
#   printf "Command line switch: list filename only\n";
  }
  elsif (/^-b(.*)/) {
    $ext = "epx";
    $dir = "bench";
    $arg = "-b ";
#   printf "Command line switch: benchmark files\n";
  }
  elsif (/^-m(.*)/) {
    $ext = "ttx";
    $dir = "manual";
    $arg = "-m ";
#   printf "Command line switch: manual files\n";
  }
  elsif (/^-u(.*)/) {
    $ext = "pl";
    $dir = "util";
    $arg = "-u ";
#   printf "Command line switch: utility (Perl) files\n";
  }
  elsif (/^-h(.*)/) {
    $ext = "his";
    $dir = "history";
    $arg = "-h ";
#   printf "Command line switch: utility (Perl) files\n";
  }
  elsif (/^-v(.*)/) {
    $ext = "vld";
    $dir = "validate";
    $arg = "-v ";
#   printf "Command line switch: validation files\n";
  }
  elsif (/^-g(.*)/) {
    $get = "1";
    $opt ="-l";
#   printf "Command line switch: copy the files to the current folder\n";
  }
  else {
    die "ERROR: unknown switch: $_\n";
  }
}
#-------------------------------------------------------------------------------
# Check if the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  die "The EUROPLEXUS environment variable is undefined!\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS source|include directory exists
#
$epxd = "$epx\\$dir";
if ( ! -d $epxd) {
  die "The EUROPLEXUS directory: $epx\$dir does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Set up pattern
#
$pattern = $ARGV[0];                          # Get fist and only argument
#-------------------------------------------------------------------------------
# Search pattern in files
#
#
if ($get eq "1") {
  unlink "_greplist.txt";
  #system("grep -i $opt \"$pattern\" $epx\\$dir\\*.$ext > _greplist.txt");
   if ( ! system ("ls $epx\\$dir\\a*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\a*.$ext >> _greplist.txt");
   }
   if ( ! system ("ls $epx\\$dir\\b*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\b*.$ext >> _greplist.txt");
   }
   if ( ! system ("ls $epx\\$dir\\c*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\c*.$ext >> _greplist.txt");
   }
   if ( ! system ("ls $epx\\$dir\\d*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\d*.$ext >> _greplist.txt");
   }
   if ( ! system ("ls $epx\\$dir\\e*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\e*.$ext >> _greplist.txt");
   }
   if ( ! system ("ls $epx\\$dir\\f*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\f*.$ext >> _greplist.txt");
   }
   if ( ! system ("ls $epx\\$dir\\g*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\g*.$ext >> _greplist.txt");
   }
   if ( ! system ("ls $epx\\$dir\\h*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\h*.$ext >> _greplist.txt");
   }
   if ( ! system ("ls $epx\\$dir\\i*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\i*.$ext >> _greplist.txt");
   }
   if ( ! system ("ls $epx\\$dir\\j*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\j*.$ext >> _greplist.txt");
   }
   if ( ! system ("ls $epx\\$dir\\k*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\k*.$ext >> _greplist.txt");
   }
   if ( ! system ("ls $epx\\$dir\\l*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\l*.$ext >> _greplist.txt");
   }
   if ( ! system ("ls $epx\\$dir\\m*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\m*.$ext >> _greplist.txt");
   }
   if ( ! system ("ls $epx\\$dir\\n*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\n*.$ext >> _greplist.txt");
   }
   if ( ! system ("ls $epx\\$dir\\o*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\o*.$ext >> _greplist.txt");
   }
   if ( ! system ("ls $epx\\$dir\\p*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\p*.$ext >> _greplist.txt");
   }
   if ( ! system ("ls $epx\\$dir\\q*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\q*.$ext >> _greplist.txt");
   }
   if ( ! system ("ls $epx\\$dir\\r*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\r*.$ext >> _greplist.txt");
   }
   if ( ! system ("ls $epx\\$dir\\s*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\s*.$ext >> _greplist.txt");
   }
   if ( ! system ("ls $epx\\$dir\\t*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\t*.$ext >> _greplist.txt");
   }
   if ( ! system ("ls $epx\\$dir\\u*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\u*.$ext >> _greplist.txt");
   }
   if ( ! system ("ls $epx\\$dir\\v*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\v*.$ext >> _greplist.txt");
   }
   if ( ! system ("ls $epx\\$dir\\w*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\w*.$ext >> _greplist.txt");
   }
   if ( ! system ("ls $epx\\$dir\\x*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\x*.$ext >> _greplist.txt");
   }
   if ( ! system ("ls $epx\\$dir\\y*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\y*.$ext >> _greplist.txt");
   }
   if ( ! system ("ls $epx\\$dir\\z*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\z*.$ext >> _greplist.txt");
   }
   open (OFILE, "_greplist.txt");
   while (<OFILE>) {
     chop;
     $fil = $_;
     push @FileNames, $_;
   }
   close OFILE;
   foreach $filename (@FileNames) {
     $epx1 = $epx;
     $epx1 =~ s/\\/\\\\/g;
     $filename =~ s/$epx1//;           # Remove $epx1 from string
     $filename =~ s/\\$dir\\//;        # Remove $dir from string
     system("epx_get $arg $filename");
   }
}
else {
#
# 27 October 2010 : under Windows 7 64-bit grep on EUROPLEXUS sources
# fails because there are "too many open files".
# To circumvent the problem, I seach letter-by-letter.
#
# The following if is used to avoid the grep error message if there are no
# files starting with a certain letter
#
   if ( ! system ("ls $epx\\$dir\\a*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\a*.$ext");
   }
   if ( ! system ("ls $epx\\$dir\\b*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\b*.$ext");
   }
   if ( ! system ("ls $epx\\$dir\\c*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\c*.$ext");
   }
   if ( ! system ("ls $epx\\$dir\\d*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\d*.$ext");
   }
   if ( ! system ("ls $epx\\$dir\\e*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\e*.$ext");
   }
   if ( ! system ("ls $epx\\$dir\\f*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\f*.$ext");
   }
   if ( ! system ("ls $epx\\$dir\\g*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\g*.$ext");
   }
   if ( ! system ("ls $epx\\$dir\\h*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\h*.$ext");
   }
   if ( ! system ("ls $epx\\$dir\\i*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\i*.$ext");
   }
   if ( ! system ("ls $epx\\$dir\\j*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\j*.$ext");
   }
   if ( ! system ("ls $epx\\$dir\\k*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\k*.$ext");
   }
   if ( ! system ("ls $epx\\$dir\\l*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\l*.$ext");
   }
   if ( ! system ("ls $epx\\$dir\\m*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\m*.$ext");
   }
   if ( ! system ("ls $epx\\$dir\\n*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\n*.$ext");
   }
   if ( ! system ("ls $epx\\$dir\\o*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\o*.$ext");
   }
   if ( ! system ("ls $epx\\$dir\\p*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\p*.$ext");
   }
   if ( ! system ("ls $epx\\$dir\\q*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\q*.$ext");
   }
   if ( ! system ("ls $epx\\$dir\\r*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\r*.$ext");
   }
   if ( ! system ("ls $epx\\$dir\\s*.$ext >_devnull 2>&1")) {
     system("grep -i $opt \"$pattern\" $epx\\$dir\\s*.$ext");
   }
}
```

```
    if ( ! system ("ls $epx\\$dir\\t*.$ext >_devnull 2>&1")) {
      system("grep -i $opt \"$pattern\" $epx\\$dir\\t*.$ext");
    }
    if ( ! system ("ls $epx\\$dir\\u*.$ext >_devnull 2>&1")) {
      system("grep -i $opt \"$pattern\" $epx\\$dir\\u*.$ext");
    }
    if ( ! system ("ls $epx\\$dir\\v*.$ext >_devnull 2>&1")) {
      system("grep -i $opt \"$pattern\" $epx\\$dir\\v*.$ext");
    }
    if ( ! system ("ls $epx\\$dir\\w*.$ext >_devnull 2>&1")) {
      system("grep -i $opt \"$pattern\" $epx\\$dir\\w*.$ext");
    }
    if ( ! system ("ls $epx\\$dir\\x*.$ext >_devnull 2>&1")) {
      system("grep -i $opt \"$pattern\" $epx\\$dir\\x*.$ext");
    }
    if ( ! system ("ls $epx\\$dir\\y*.$ext >_devnull 2>&1")) {
      system("grep -i $opt \"$pattern\" $epx\\$dir\\y*.$ext");
    }
    if ( ! system ("ls $epx\\$dir\\z*.$ext >_devnull 2>&1")) {
      system("grep -i $opt \"$pattern\" $epx\\$dir\\z*.$ext");
    }
}

#
exit;
```

epx_hist.pl

```
#-------------------------------------------------------------------------------
# Extract from the EUROPLEXUS global history file the info relative to a file
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$his = "E:\\EUROPLEXUS\\epx-history\.txt";   # Name of the global history file
$errfil = "epx_hist.err";                    # Name of the error file
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 1
#
if ($#ARGV != 0) {                       # Index of last argument must be = 0
  $errmsg = "Usage: epx_hist filename\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS global history file exists
#
if ( ! -r $his ) {
  $errmsg = "The EUROPLEXUS history file: $his does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# The argument is the file name (without extension!)
#
$file = $ARGV[0];
$file =~ tr/A-Z/a-z/;
#-------------------------------------------------------------------------------
# Open the global history file and extract (print to STDOUT) all the
# information relative to the given file name
#
print "Extracting from global history file: $his\n";
print "the information relative to file   : $file.\n";
print "================================================\n";
#
$echo = 0;
open (HIST, $his);
while (<HIST>) {
  chop;
  if ( m/^[\*\$\%C] .*\x2F..\x2F.....:.:.. *$/ ) {
    #-----------------------------------------------------------------------
    # Line contains "old" header (without evolution number at the end):
    #   - Extract file name (2nd line field) and copy file contents
    #     up to next blank line to STDOUT
    #
    ($comm,$name,$status,$owner,$date,$time) = split(/[ \t]+/, $_);
    $name =~ tr/A-Z/a-z/;
    if ($name eq $file) {
      $echo = 1;
      print "\n", $_, "\n";
    }
  }
  if ( m/^[\*\$\%C] .*\x2F..\x2F.....:.:.. #\d\d\d\d *$/ ) {
    #-----------------------------------------------------------------------
    # Line contains "new" header (with evolution number at the end):
    #   - Extract file name (2nd line field) and copy file contents
    #     up to next blank line to STDOUT
    #
    ($comm,$name,$status,$owner,$date,$time,$evo) = split(/[ \t]+/, $_);
    $name =~ tr/A-Z/a-z/;
    if ($name eq $file) {
      $echo = 1;
      print "\n", $_, "\n";
    }
  }
  elsif ( m/\S+/ ) {
    #
    # Non-header, non-blank line
    #
    if ($echo) {
      print $_, "\n";
    }
  }
  else {
    #
    # Blank line
    #
    $echo = 0;
  }
}
close HIST;
#
exit;
```

```
#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_init.pl

```
#-------------------------------------------------------------------------------
# Initialise EUROPLEXUS development Solution
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$copy = "cp -p";                      # Alternative: use copy
$sln = "plex.sln";                    # Solution file
$ncb = "plex.ncb";                    # NCB file
$suo = "plex.suo";                    # OPT file
$d_filter = "filter";                 # filter project sub-directory
$p_filter = "filter.vcproj";          # filter project file (filtering)
$d_epx = "epx";                       # epx project sub-directory
$p_epx = "epx.vfproj";                # epx project file (compilation)
$main = "main.ff";                    # main source file
$rule = "ff_rule.rules";              # custom rule file for .ff extension
$pfad1 = "filter1.txt";               # First part of project for filter
$pfad2 = "filter2.txt";               # Second part of project for filter
$pfad3 = "epx1.txt";                  # First part of project for epx
$pfad4 = "epx2.txt";                  # Second part of project for epx
#-------------------------------------------------------------------------------
# Check number of arguments, must be 0 or 1
#
if ($#ARGV >= 2) {                     # Index of last argument must be -1 or 0
  die "Usage: epx_init [-w] [-e] [-d] [-?]\007\n";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#       -w      generate QuickWin Solution instead of Console
#
while ($ARGV[0] =~ /^-/) {
  $_ = shift;
  if (/^-\?(.*)/) {
    printf "Initialise EUROPLEXUS development Solution\n";
    printf "-w QuickWin\n";
    printf "-e Empty project files - no files are added to the project\n";
    printf "-d Delete the solution files before initializing\n";
    printf "-? Print this help\n";
    exit;
  }
  elsif (/^-w(.*)/) {
    $qw = "QuickWin";
    printf "Command line switch: QuickWin\n";
  }
  elsif (/^-e(.*)/) {
    $empty = "1";
    printf "Command line switch: Empty project files\n";
  }
  elsif (/^-d(.*)/) {
    $delete = "1";
    printf "Delete the solution files before initializing\n";
  }
  else {
    $errmsg = "ERROR: unknown switch: $_\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
}
#-------------------------------------------------------------------------------
# Check if the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  die "The EUROPLEXUS environment variable is undefined!\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  die "The EUROPLEXUS directory: $epx does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS init subdirectory exists
#
if (defined $qw) {
  $initdir = "$epx\\init_quickwin";
}
else {
  $initdir = "$epx\\init";
}
if ( ! -d $initdir) {
  die "The EUROPLEXUS init directory: $initdir does not exist!\007\n";
}
# Delete existing files
if ( $delete =="1"){
  if ( -r $sln ) {
    system("rm $sln");
  }
  if ( -r $ncb ) {
    system("rm $ncb");
  }
  if ( -r $suo ) {
    system("rm $suo");
  }
  if ( -r $pfad1 ) {
    system("rm $pfad1");
  }
  if ( -r $pfad2 ) {
    system("rm $pfad2");
  }
  if ( -r $pfad3 ) {
```

```
    system("rm $pfad3");
  }
  if ( -r $pfad4 ) {
    system("rm $pfad4");
  }
  if ( -r $rule ) {
    system("rm $rule");
  }
  system("rm -f epx_ff.txt");
  system("rm -f epx_ordo.txt");
  if ( -d $d_filter ) {
    system("rm -f $d_filter/Debug/*.*");
    system("rmdir $d_filter/Debug");
    system("rm -f $d_filter/*.*");
    system("rmdir $d_filter");
  }
  if ( -d $d_epx ) {
    system("rm -f $d_epx/Debug/*.*");
    system("rmdir $d_epx/Debug");
    system("rm -f $d_epx/*.*");
    system("rmdir $d_epx");
  }
}
#-------------------------------------------------------------------------------
# Verify that the files to be copied do not exist already in "."
#
if ( -r $sln ) {                          # Test if target exists already
  die "File $sln exists already in current directory!\007\n";
}
if ( -r $ncb ) {                          # Test if target exists already
  die "File $ncb exists already in current directory!\007\n";
}
if ( -r $suo ) {                          # Test if target exists already
  die "File $suo exists already in current directory!\007\n";
}
if ( -d $d_filter ) {                     # Test if target exists already
  die "Sub-directory $d_filter exists already in current directory!\007\n";
}
if ( -d $d_epx ) {                        # Test if target exists already
  die "Sub-directory $d_epx exists already in current directory!\007\n";
}
if ( -r $rule ) {                         # Test if target exists already
  die "File $rule exists already in current directory!\007\n";
}
if ( $empty !="1"){
  if ( -r $pfad1 ) {                        # Test if target exists already
    die "File $pfad1 exists already in current directory!\007\n";
  }
  if ( -r $pfad2 ) {                        # Test if target exists already
    die "File $pfad2 exists already in current directory!\007\n";
  }
  if ( -r $pfad3 ) {                        # Test if target exists already
    die "File $pfad3 exists already in current directory!\007\n";
  }
  if ( -r $pfad4 ) {                        # Test if target exists already
    die "File $pfad4 exists already in current directory!\007\n";
  }
}
#-------------------------------------------------------------------------------
$copymain = 1;
if ( -r $main ) {                         # Test if target exists already
  print "File $main is already here, it won't be copied.\007\n";
  undef $copymain;
}
if (defined $qw) {
  $copymqw = 1;
  if ( -r m_qwin.ff ) {                   # Test if target exists already
    print "File m_qwin.ff is already here, it won't be copied.\007\n";
    undef $copymqw;
  }
  $copyifw = 1;
  if ( -r ifwin.ff ) {                    # Test if target exists already
    print "File ifwin.ff is already here, it won't be copied.\007\n";
    undef $copyifw;
  }
}
#-------------------------------------------------------------------------------
# Verify that the relevant files exist in the EUROPLEXUS init directory
#
$from = "$initdir\\$sln";                 # Full file name
if ( ! -r $from ) {
  die "File $sln does not exist in $initdir!\007\n";
}
$from = "$initdir\\$ncb";                 # Full file name
if ( ! -r $from ) {
  die "File $ncb does not exist in $initdir!\007\n";
}
$from = "$initdir\\$suo";                 # Full file name
if ( ! -r $from ) {
  die "File $suo does not exist in $initdir!\007\n";
}
$from = "$initdir\\$d_filter\\$p_filter"; # Full file name
if ( ! -r $from ) {
  die "File $d_filter\\$p_filter does not exist in $initdir!\007\n";
}
$from = "$initdir\\$d_epx\\$p_epx";       # Full file name
if ( ! -r $from ) {
  die "File $d_epx\\$p_epx does not exist in $initdir!\007\n";
}
$from = "$initdir\\$rule";                # Full file name
if ( ! -r $from ) {
  die "File $rule does not exist in $initdir!\007\n";
}
if ( $empty !="1"){
  $from = "$initdir\\$pfad1";               # Full file name
  if ( ! -r $from ) {
    die "File $pfad1 does not exist in $initdir!\007\n";
  }
  $from = "$initdir\\$pfad2";               # Full file name
  if ( ! -r $from ) {
    die "File $pfad2 does not exist in $initdir!\007\n";
  }
  $from = "$initdir\\$pfad3";               # Full file name
  if ( ! -r $from ) {
    die "File $pfad3 does not exist in $initdir!\007\n";
  }
  $from = "$initdir\\$pfad4";               # Full file name
  if ( ! -r $from ) {
    die "File $pfad4 does not exist in $initdir!\007\n";
  }
}
#-------------------------------------------------------------------------------
# Copy files from the EUROPLEXUS init to the current directory
#
$from = "$initdir\\$sln";                 # Full file name
system("$copy $from .");
$from = "$initdir\\$ncb";                 # Full file name
system("$copy $from .");
$from = "$initdir\\$suo";                 # Full file name
system("$copy $from .");
$from = "$initdir\\$d_filter\\$p_filter"; # Full file name
system("mkdir $d_filter");
if ( $empty =="1"){  # copy standard file
  system("$copy $from $d_filter");
}
$from = "$initdir\\$d_epx\\$p_epx";       # Full file name
system("mkdir $d_epx");
if ( $empty =="1"){  # copy standard file
  system("$copy $from $d_epx");
}
$from = "$initdir\\$rule";                # Full file name
system("$copy $from .");
if ( $empty !="1"){
  $from = "$initdir\\$pfad1";               # Full file name
  system("$copy $from .");
  $from = "$initdir\\$pfad2";               # Full file name
  system("$copy $from .");
  $from = "$initdir\\$pfad3";               # Full file name
  system("$copy $from .");
  $from = "$initdir\\$pfad4";               # Full file name
  system("$copy $from .");
}
print "Solution files were copied to \"\.\".\n";

#-------------------------------------------------------------------------------
# Get the main source file if not already present
#
if (defined ($copymain)) {
  system("epx_get $main");
}

#
#Create the project files with the fortran files
#
if ( $empty !="1"){
  $ntot = `ls *.ff | wc -l`;       # number of .ff files in current directory
  chop $ntot;
  print "There are $ntot source files to insert\n";

  $outfile = "epx_ff.txt";
  $com_oth = "grep -v \"\*\.ff\"";
  system ("ls *.ff | $com_oth >> $outfile");
#-------------------------------------------------------------------------------
# Build array of source file names from file epx_ff.txt without main.ff
#
  open (OFILE, $outfile);
  while (<OFILE>) {
    chop;
    $fil = $_;
    if(index($_,"main.ff")==-1){
       push @FileNames, $_;
    }
  }
  close OFILE;

#creating of the filter-file
  $pfad ="$d_filter\\$p_filter";
  open(FILTER, ">$pfad");

#template-datei no. 1 for the filter
  open(DATEI, $pfad1);
  @inhalt=<DATEI>;

  foreach $zeile (@inhalt) {
    print FILTER "$zeile";
  }
  close(DATEI);

#including the ff-files
  foreach $filename (@FileNames) {
    print FILTER "<File\n";
    print FILTER "RelativePath=\"..\\";
    print FILTER $filename;
    print FILTER "\"\n>\n";
    print FILTER "</File>\n";
  }

#template-datei no. 2 for the filter
  open(DATEI2, $pfad2);
  @inhalt=<DATEI2>;

  foreach $zeile (@inhalt) {
    print FILTER "$zeile";
  }
  close(DATEI2);
  close(FILTER);

#also for the epx-file
  $pfad ="$d_epx\\$p_epx";
  open(FILTER, ">$pfad");

#template-datei no. 1 for the epx
  open(DATEI, $pfad3);
  @inhalt=<DATEI>;

  foreach $zeile (@inhalt) {
    print FILTER "$zeile";
  }
  close(DATEI);

#including the f-files
  foreach $filename (@FileNames) {
    chop($filename);
    print FILTER "<File RelativePath=\"..\\";
    print FILTER $filename;
    print FILTER "\"/>\n";
```

```
  }

#template-datei no. 2 for the epx
  open(DATEI2, $pfad4);
  @inhalt=<DATEI2>;

  foreach $zeile (@inhalt) {
    print FILTER "$zeile";
  }
  close(DATEI2);
  close(FILTER);
}
#-------------------------------------------------------------------------------
# Get the QuickWin-dependent source files if not already present
#
if (defined ($copymqw)) {
  system("epx_get m_qwin");
}
if (defined ($copyifw)) {
  system("epx_get ifwin");
}
#
exit;


#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_lk.pl

```
#-------------------------------------------------------------------------------
# Link and produce local EUROPLEXUS executable
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$out = "epx.exe";
$deb = "/DEBUG";
$libs = "/SUBSYSTEM:console";
#$nodef = "";
$nodef = "/NODEFAULTLIB:libc.lib";
$force = "";
$libf = "Libplex.lib";
$libf_mpi = "Libplex_mpi.lib";
$libf_check = "Libplex_check.lib";
$libb = "Libblas.lib";
$liblap= "Liblapack.lib";
$libsp= "Libsp.lib";
$libogl = "f90gl.lib f90glu.lib f90glut.lib glut32.lib bmplib.lib user32.lib gdi
32.lib vfw32.lib";
$errfil = "epx_lk.err";
$profmap = "";
$stack = "0x640000"; # 10 MB by default
#-------------------------------------------------------------------------------
# Check number of arguments, must be 0 or 1 or 2 or 3 or 4 or 5 or 6 or 7
#
if ($#ARGV > 6) {                          # Index of last argument must be <= 6
  $errmsg = "Usage: epx_lk [-l|-L] [-o] [-w] [-c] [-f] [-p] [-e NAME] [-s] [-M]
[-nomkl]\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#       -l      generate and use local library by copying standard library
#               and modifying it with local object files
#       -L      generate and use complete local library, ignore standard one
#               (for use with evolution involving full compilation)
#               assumes that all sources have been compiled locally
#       -o      optimize (no PDB debug info)
#       -w      generate QuickWin graphic application instead of console
#       -c      link against library compiled with -check
#       -f      force executable creation even in case of link errors
#       -p      generate map for profiling
#       -e NAME use NAME as name of the executable instead of epx.exe
#       -s SIZE use SIZE as stack size (in the form 0xnnnnn)
#       -M      link for MPI (parallel) version
#       -nomkl  do not use Intel MKL libraries (use local Blas/Lapack)
#
while ($ARGV[0] =~ /^-/) {
  $_ = shift;
  if (/^-o(.*)/) {
    $deb = "";
    printf "Command line switch: optimize (no PDB debug info)\n";
  }
  elsif (/^-l(.*)/) {
    $loc = $libf;
    printf "Command line switch: local library\n";
  }
  elsif (/^-L(.*)/) {
    $loc = $libf;
    $full_compil = "yes";
    printf "Command line switch: local library (full compilation)\n";
  }
  elsif (/^-w(.*)/) {
    $qw = "QuickWin";
    $out = "epxqw.exe";
    $libs = "/SUBSYSTEM:windows";
#   $nodef = "/NODEFAULTLIB:libc.lib";
    printf "Command line switch: QuickWin\n";
  }
  elsif (/^-c(.*)/) {
    $check = "yes";
    printf "Command line switch: use check library\n";
  }
  elsif (/^-f(.*)/) {
    $force = "/FORCE";
    printf "Command line switch: force\n";
  }
  elsif (/^-p(.*)/) {
    $profmap = "/MAP:epx.map";
#   $profmap = "/PROFILE";
    printf "Command line switch: Map\n";
  }
  elsif (/^-e$/) {
    $_ = shift;
    $out = $_;
    printf "Command line switch: name of the executable: $out\n";
  }
  elsif (/^-s$/) {
    $_ = shift;
    $stack = $_;
    printf "Command line switch: use stack size $stack\n";
  }
  elsif (/^-M$/) {
    $libmpi = "mpi.lib fmpich2.lib";
    printf "Command line switch: link for MPI\n";
  }
  elsif (/^-nomkl$/) {
    $nomkl = "yes";
    printf "Command line switch: do not use Intel MKL libraries\n";
  }
  else {
    $errmsg = "ERROR: unknown switch: $_\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
}
#-------------------------------------------------------------------------------
# Optional switches -c and -M are mutually exclusive
if (defined($check)) {
  if (defined($libmpi)) {
    $errmsg = "-c and -M are mutually exclusive!\007\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
}
#-------------------------------------------------------------------------------
# Optional switches -c and -w are mutually exclusive
if (defined($check)) {
  if (defined($qw)) {
    $errmsg = "-c and -w are mutually exclusive!\007\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
}
#-------------------------------------------------------------------------------
# Optional switches -w and -M are mutually exclusive
if (defined($qw)) {
  if (defined($libmpi)) {
    $errmsg = "-w and -M are mutually exclusive!\007\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
}
#-------------------------------------------------------------------------------
# If the -M switch was activated (MPI version), use the MPI version of
# the library !!!
#
if (defined($libmpi)) {
  if (defined($loc) ) {
    $loc = $libf_mpi;
  }
  $libf = $libf_mpi;
}
else {
  $libmpi = "";
}
#-------------------------------------------------------------------------------
# If the -c switch was activated (-check version), use the _check version of
# the library !!!
#
if (defined($check)) {
  if (defined($loc) ) {
    $loc = $libf_check;
  }
  $libf = $libf_check;
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  $errmsg = "The EUROPLEXUS environment variable is undefined!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  $errmsg = "The EUROPLEXUS directory: $epx does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS library directory exists
#
$libdir = "$epx\\library";
if ( ! -d $libdir) {
  $errmsg = "The EUROPLEXUS library directory: $libdir does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS library files exists (except in case -L)
#
if ( ! defined ($full_compil)) {
  $lib1 = "$libdir\\$libf";  # use standard library
}
else {
  $lib1 = "$libf";           # ignore standard library
}
if ( ! defined ($full_compil)) {
  if ( ! -r $lib1) {
    $errmsg = "The EUROPLEXUS library file: $lib1 does not exist!\007\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
}
if (defined($nomkl)) {
  $lib3 = "$libdir\\$libb";
  if ( ! -r $lib3) {
```

```
    $errmsg = "The EUROPLEXUS library file: $lib3 does not exist!\007\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
  $lib4 = "$libdir\\$liblap";
  if ( ! -r $lib4) {
    $errmsg = "The EUROPLEXUS library file: $lib4 does not exist!\007\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
}
$lib5 = "$libdir\\$libsp";
if ( ! -r $lib5) {
  $errmsg = "The EUROPLEXUS library file: $lib5 does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Remove the executable if it exists already
#
system("rm -f $out");
#-------------------------------------------------------------------------------
if (defined($loc)) {
  #
  # Local library switch (-l or -L) has been specified:
  #
  # Remove the local library if it exists already
  system ("rm -f $loc");
  if ( ! defined ($full_compil)) {
    # copy standard library here and update it by local objects
    system ("cp -p $lib1 $loc");
    system ("LIB $loc *.obj");
  }
  else {
    # full recompilation : build the library here ex novo
    system ("LIB /OUT:$loc *.obj");
  }
  #
  # Redefine $lib1 so that (in any case) it points to local library
  #
  $lib1 = $loc;
  $objects = "main.obj";
}
else {
  $objects = "*.obj";
}
if (defined ($qw)) {
  #-----------------------------------------------------------------------------
  # QuickWin version requires compilation of IFWIN.ff and M_QWIN.ff with
  # additional QWIN switch!
  # Intel 9.0 compiler: requires also extraction of MAIN.ff and compilation
  # of ALL *.ff FILES IN CURRENT DIR with /libs:qwin (epx_cmp -g ...) !!!
  #-----------------------------------------------------------------------------
  # Retrieve ifwin.ff from sources if not existing already in the current dir
  #
  $ifwin="ifwin.ff";
  if ( ! -r $ifwin ) {
    print "Retrieving ifwin.ff from sources!\n";
    #
    if ( system ("epx_get ifwin")) {
      $errmsg = "Unable to retrieve ifwin.ff from sources!\n";
      &ERRFIL ($errfil, $errmsg); die "$errmsg";
    }
    #
    $remifw = "rm -f ifwin.*";
    print "Define remifw: $remifw \n";
  }
  # Retrieve m_qwin.ff from sources if not existing already in the current dir
  #
  $m_qwin="m_qwin.ff";
  if ( ! -r $m_qwin ) {
    print "Retrieving m_qwin.ff from sources!\n";
    #
    if ( system ("epx_get m_qwin")) {
      $errmsg = "Unable to retrieve m_qwin.ff from sources!\n";
      &ERRFIL ($errfil, $errmsg); die "$errmsg";
    }
    #
    $remmqw = "rm -f m_qwin.*";
    print "Define remmqw: $remmqw \n";
  }
  # Retrieve main.ff from sources if not existing already in the current dir
  #
  $mainff="main.ff";
  if ( ! -r $mainff ) {
    print "Retrieving main.ff from sources!\n";
    #
    if ( system ("epx_get main")) {
      $errmsg = "Unable to retrieve main.ff from sources!\n";
      &ERRFIL ($errfil, $errmsg); die "$errmsg";
    }
    #
    $remmain = "rm -f main.ff main.f main.err";
    print "Define remmain: $remmain \n";
  }
  # (Re-)compile ALL SOURCES IN CURRENT DIR with -g SWITCH !!!
  #
  print "ATTENTION: (Re-)compiling all current sources with -g !!!\n";
  #
  if ( system ("epx_cmp -o -g")) {
    $errmsg = "Unable to compile all current sources with -g!\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
  $remallobj = "rm -f *.obj";
  # Re-compile m_qwin and ifwin with additional QWIN keyword and with -g
  #
  print "Re-compiling m_qwin and ifwin with the QWIN keyword and -g\n";
  #
  if ( system ("epx_cmp -o -g -k QWIN m_qwin ifwin")) {
    $errmsg = "Unable to re-compile m_qwin and ifwin!\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
  $objects = "*.obj";
}
else {
  #-----------------------------------------------------------------------------
  # NON-QuickWin version:
  #-----------------------------------------------------------------------------
  # Extract main.obj from lib1 if not existing already in the current dir
  #
  $main = "main.obj";
  if ( ! -r $main ) {
    print "Extracting main.obj from library $lib1!\n";
    #
    if ( system ("LIB /EXTRACT:main.obj $lib1")) {
      $errmsg = "Unable to extract main.obj from $lib1!\n";
      &ERRFIL ($errfil, $errmsg); die "$errmsg";
    }
    #
    $remmainobj = "rm -f main.obj";
  }
#-------------------------------------------------------------------------------
} # End if QuickWin version
#-------------------------------------------------------------------------------
# Link and produce the executable
#
if (defined($nomkl)) {
  $libr = "$libf $libb $liblap $libsp $libogl $libmpi";
}
else {
  $libr = "$libf $libsp $libogl $libmpi";
}
#
#print "Library 1: $libf\n";
#print "Library 3: $libb\n";
#print "Library 4: $liblap\n";
#print "Library 5: $libsp\n";
#
print "LINK /OUT:$out /INCREMENTAL:NO /LIBPATH:\"$libdir\" /STACK:$stack\n";
print "      $deb $force $profmap $libs $nodef\n";
print "      $objects $libr\n";
#
if (system("LINK /OUT:$out /INCREMENTAL:NO /LIBPATH:\"$libdir\" /STACK:$stack
                $deb $force $profmap $libs $nodef
                $objects $libr")) {
  $errmsg = "Link ERROR(s)!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Remove main.obj (but only if it was previously extracted from lib1)
#
if (defined ($remmainobj)) {
  print ("Deleting main.obj!\n");
  system("$remmainobj");
}
#-------------------------------------------------------------------------------
# Remove all object files in case this is a QuickWin version
# (since they have all been re-compiled with the -g option)
#
if (defined ($remallobj)) {
  print ("Deleting all object files (were recompiled with -g)!\n");
  system("$remallobj");
}
#-------------------------------------------------------------------------------
# Clean up main.ff main.f and main.err if necessary
#
if (defined ($remmain)) {
  print ("Deleting main.ff main.f main.err!\n");
  system("$remmain");
}
#-------------------------------------------------------------------------------
# Remove ifwin.* (but only if it was previously extracted from source)
#
if (defined ($remifw)) {
  print ("Deleting ifwin.*!\n");
  system("$remifw");
}
#-------------------------------------------------------------------------------
# Remove m_qwin.* (but only if it was previously extracted from source)
#
if (defined ($remmqw)) {
  print ("Deleting m_qwin.*!\n");
  system("$remmqw");
}
#-------------------------------------------------------------------------------
print "The local executable: $out has been produced.\n";
#
exit;


#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_mail.bat

```
@rem = '--*-Perl-*--
@echo off
if "%OS%" == "Windows_NT" goto WinNT
perl -x -S "%0" %1 %2 %3 %4 %5 %6 %7 %8 %9
goto endofperl
:WinNT
perl -x -S %0 %*
if NOT "%COMSPEC%" == "%SystemRoot%\system32\cmd.exe" goto endofperl
if %errorlevel% == 9009 echo You do not have Perl in your PATH.
if errorlevel 1 goto script_failed_so_exit_with_non_zero_val 2>nul
goto endofperl
@rem ';
#!perl
#line 15
#-------------------------------------------------------------------------------
# Send file by e-mail in batch mode
# Calls epx_mail_1 once for each recipient
#-------------------------------------------------------------------------------
use Mail::Sendmail; # This version uses SendMail 2.09 (NOT 0.78!)
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
```

```
#
#$from = "<folco.casadei\@jrc.ec.europa.eu>";
#$to   = "<hbung\@cea.fr>";
#$cc   = "<folco.casadei\@jrc.ec.europa.eu>, <eros.gabellini\@samtech.fr>, <vinc
ent.faucher\@cea.fr>, <serguei.potapov\@edf.fr>, <jeanfrancois.marechal\@samcef.
com>, <dinh\@samcef.com>, <georgios.valsamos\@jrc.ec.europa.eu>";
$errfil = "epx_mail.err";
#-------------------------------------------------------------------------------
# Check number of arguments, must be 3
#
if ($#ARGV != 2 ) {                          # Index of last argument must be 2
  $errmsg = "Usage: epx_mail status num filename\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Set up file name
#
$stat = $ARGV[0];                            # First argument: evolution status
                                             # "OK" or "FAILED!!!"
$enum = $ARGV[1];                            # Second argument: evolution number
$file = $ARGV[2];                            # Third argument: log file name
#
$sbj  = "EUROPLEXUS evolution \# $enum at JRC: $stat";
#-------------------------------------------------------------------------------
# Verify that the file exists
#
if ( ! -r $file ) {
  $errmsg = "File $file does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Call epx_mail_1 once for each recipient, since sendmail 2.09 does not
# seem to accept more than one recipient
#
$rec = "folco.casadei\@jrc.ec.europa.eu";
if (system ("epx_mail_1 $stat $enum $file $rec")) {
  $errmsg = "Mail ERROR(S) in sending to $rec.\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
$rec = "hbung\@cea.fr";
if (system ("epx_mail_1 $stat $enum $file $rec")) {
  $errmsg = "Mail ERROR(S) in sending to $rec.\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
$rec = "vincent.faucher\@cea.fr";
if (system ("epx_mail_1 $stat $enum $file $rec")) {
  $errmsg = "Mail ERROR(S) in sending to $rec.\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
$rec = "serguei.potapov\@edf.fr";
if (system ("epx_mail_1 $stat $enum $file $rec")) {
  $errmsg = "Mail ERROR(S) in sending to $rec.\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
$rec = "dinh.le\@samtech.com";
if (system ("epx_mail_1 $stat $enum $file $rec")) {
  $errmsg = "Mail ERROR(S) in sending to $rec.\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
$rec = "roland.ortiz\@onera.fr";
if (system ("epx_mail_1 $stat $enum $file $rec")) {
  $errmsg = "Mail ERROR(S) in sending to $rec.\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
$rec = "georgios.valsamos\@jrc.ec.europa.eu";
if (system ("epx_mail_1 $stat $enum $file $rec")) {
  $errmsg = "Mail ERROR(S) in sending to $rec.\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
$rec = "casadeifolco\@gmail.com";
if (system ("epx_mail_1 $stat $enum $file $rec")) {
  $errmsg = "Mail ERROR(S) in sending to $rec.\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
$rec = "alberto.beccantini\@cea.fr";
if (system ("epx_mail_1 $stat $enum $file $rec")) {
  $errmsg = "Mail ERROR(S) in sending to $rec.\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#
print "Done\n\n";
exit 0;

#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------

__END__
:endofperl
```

epx_mail.pl

```
#-------------------------------------------------------------------------------
# Send file by e-mail in batch mode
# Calls epx_mail_1 once for each recipient
#-------------------------------------------------------------------------------
use Mail::Sendmail; # This version uses SendMail 2.09 (NOT 0.78!)
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
#$from = "<folco.casadei\@jrc.ec.europa.eu>";
#$to   = "<hbung\@cea.fr>";
#$cc   = "<folco.casadei\@jrc.ec.europa.eu>, <eros.gabellini\@samtech.fr>, <vinc
ent.faucher\@cea.fr>, <serguei.potapov\@edf.fr>, <jeanfrancois.marechal\@samcef.
com>, <dinh\@samcef.com>, <georgios.valsamos\@jrc.ec.europa.eu>";
$errfil = "epx_mail.err";
#-------------------------------------------------------------------------------
# Check number of arguments, must be 3
#
if ($#ARGV != 2 ) {                          # Index of last argument must be 2
  $errmsg = "Usage: epx_mail status num filename\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Set up file name
#
$stat = $ARGV[0];                            # First argument: evolution status
                                             # "OK" or "FAILED!!!"
$enum = $ARGV[1];                            # Second argument: evolution number
$file = $ARGV[2];                            # Third argument: log file name
#
$sbj  = "EUROPLEXUS evolution \# $enum at JRC: $stat";
#-------------------------------------------------------------------------------
# Verify that the file exists
#
if ( ! -r $file ) {
  $errmsg = "File $file does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Call epx_mail_1 once for each recipient, since sendmail 2.09 does not
# seem to accept more than one recipient
#
$rec = "folco.casadei\@jrc.ec.europa.eu";
if (system ("epx_mail_1 $stat $enum $file $rec")) {
  $errmsg = "Mail ERROR(S) in sending to $rec.\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
$rec = "hbung\@cea.fr";
if (system ("epx_mail_1 $stat $enum $file $rec")) {
  $errmsg = "Mail ERROR(S) in sending to $rec.\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
$rec = "vincent.faucher\@cea.fr";
if (system ("epx_mail_1 $stat $enum $file $rec")) {
  $errmsg = "Mail ERROR(S) in sending to $rec.\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
$rec = "serguei.potapov\@edf.fr";
if (system ("epx_mail_1 $stat $enum $file $rec")) {
  $errmsg = "Mail ERROR(S) in sending to $rec.\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
$rec = "dinh.le\@samtech.com";
if (system ("epx_mail_1 $stat $enum $file $rec")) {
  $errmsg = "Mail ERROR(S) in sending to $rec.\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
$rec = "roland.ortiz\@onera.fr";
if (system ("epx_mail_1 $stat $enum $file $rec")) {
  $errmsg = "Mail ERROR(S) in sending to $rec.\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
$rec = "georgios.valsamos\@jrc.ec.europa.eu";
if (system ("epx_mail_1 $stat $enum $file $rec")) {
  $errmsg = "Mail ERROR(S) in sending to $rec.\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
$rec = "casadeifolco\@gmail.com";
if (system ("epx_mail_1 $stat $enum $file $rec")) {
  $errmsg = "Mail ERROR(S) in sending to $rec.\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
$rec = "alberto.beccantini\@cea.fr";
if (system ("epx_mail_1 $stat $enum $file $rec")) {
  $errmsg = "Mail ERROR(S) in sending to $rec.\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#
print "Done\n\n";
exit 0;

#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_mail_1.pl

```
#-------------------------------------------------------------------------------
# Send file by e-mail in batch mode to a single recipient
# (because sendmail 2.09 does not seem to accept multiple recipients)
#-------------------------------------------------------------------------------
use Mail::Sendmail; # This version uses SendMail 2.09 (NOT 0.78!)
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$from = "<folco.casadei\@jrc.ec.europa.eu>";
#$to   = "<hbung\@cea.fr>";
#$cc   = "<folco.casadei\@jrc.ec.europa.eu>, <eros.gabellini\@samtech.fr>, <vinc
ent.faucher\@cea.fr>, <serguei.potapov\@edf.fr>, <jeanfrancois.marechal\@samcef.
com>, <dinh\@samcef.com>, <georgios.valsamos\@jrc.ec.europa.eu>";
$errfil = "epx_mail.err";
#-------------------------------------------------------------------------------
# Check number of arguments, must be 4
#
if ($#ARGV != 3 ) {                          # Index of last argument must be 3
  $errmsg = "Usage: epx_mail status num filename recipient\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Set up file name
#
```

```
$stat = $ARGV[0];                       # First argument: evolution status
                                        # "OK" or "FAILED!!!"
$enum = $ARGV[1];                       # Second argument: evolution number
$file = $ARGV[2];                       # Third argument: log file name
$to = $ARGV[3];                         # Fourth argument: recipient
#
$sbj  = "EUROPLEXUS evolution \# $enum at JRC: $stat";
#-------------------------------------------------------------------------------
# Verify that the file exists
#
if ( ! -r $file ) {
  $errmsg = "File $file does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
#print "Stat=$stat.\n";
#print "Enum=$enum.\n";
#print "File=$file.\n";
#print "To  =$to.\n";
#-------------------------------------------------------------------------------
# Prepare message
#
$msg = "============================================================\n";
$msg = $msg."This message is automatically sent by a perl script.\n";
$msg = $msg."Attached is the trace file of an EUROPLEXUS evolution at JRC.\n";
$msg = $msg."================================================================\n\n";
#-------------------------------------------------------------------------------
# Send file by mail :
#
# Create the object with arguments,
# i.e. the machine where this script is run does not have a SMTP server.
#
#$sm = new SendMail("isis-ms.jrc.it");
$sm = new SendMail("ipsc-mail.jrc.it");
#
# Set SMTP AUTH login profile.
# Uncomment the following line if you like to try SMTP AUTH.
#
#$sm->setAuth($sm->AUTHPLAIN, "username", "password");
$sm->setAuth($sm->AUTHLOGIN, "casadfo", "fol1412%");
#
# We set the debug mode "ON".
#
#$sm->setDebug($sm->ON);
#
# We set the sender.
#
#$sm->From("Folco  <folco.casadei\@jrc.it>");
$sm->From("$from");
#
# We set the subject.
#
$sm->Subject("$sbj");
#
# We set the recipient(s).
#
#$sm->To("Folco  <folco.casadei\@jrc.it>");
$sm->To("$to");
#$sm->Cc("$cc");
#
# We set the content of the mail.
#
$sm->setMailBody("$msg");
#
# Attach the log file.
#
$sm->Attach("$file");
#
# Check if the mail sent successfully or not.
#
if ($sm->sendMail() != 0) {
  #print $sm->{'error'}."\n";
  #exit -1;
  $errmsg = $sm->{'error'}."\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}

#
# Mail sent successfully.
#
print "Sent to:$to.\n";
exit 0;

#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_make.pl

```
#-------------------------------------------------------------------------------
# Rebuild previously restored version of EUROPLEXUS
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  die "The EUROPLEXUS environment variable is undefined!\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  die "The EUROPLEXUS directory: $epx does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS\save directory exists
#
$logdir = "$epx\\save";
if ( ! -d $logdir) {
  die "The EUROPLEXUS directory: $logdir does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Redirect STDOUT and STDERR to make.log
#
$logfile = "$logdir\\make.log";
open STDOUT, ">$logfile";
open STDERR, ">$logfile";
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$copy = "cp -p -f";
$del = "rm -f";
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 0
#
if ($#ARGV != -1) {                     # Index of last argument must be = -1
  $errmsg = "Usage: epx_make\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
print "===> epx_make\n";
print "===> epx_make\n";
print "===> epx_make\n";
print "===> epx_make\n";
print "===> epx_make\n";
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  $errmsg = "The EUROPLEXUS directory: $epx does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check existence of evolution version file
#
$evonumfile = "$epx\\VERSION.txt";         # Evolution version file
if (! -r $evonumfile) {
  die "The evolution number file $evonumfile does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Set and check existence of auxiliary directories
#
$makdir = "$epx\\Make";                    # Work directory for making
if ( -d $makdir) {
  system("rm -r -f $makdir");
}
system("mkdir $makdir");
#-------------------------------------------------------------------------------
# Make sure we are in $makdir (so we may use . instead of $makdir)
#
print "Changing directory to $makdir.\n";
chdir "$makdir" ||
  die "Can't chdir to $makdir!\007\n";
#-------------------------------------------------------------------------------
# Set and check existence of target directories
#
$srcdir = "$epx\\source";
$sccdir = "$epx\\source_c";
$scbdir = "$epx\\blas_source";
$mafdir = "$epx\\manual_filtered";
$libdir = "$epx\\library";
$moddir = "$epx\\module";
$exedir = "$epx\\exe";
$utidir = "$epx\\util";
#
if ( ! -d $srcdir) {
  die "The EUROPLEXUS directory: $srcdir does not exist!\007\n";
}
if ( ! -d $sccdir) {
  die "The EUROPLEXUS directory: $sccdir does not exist!\007\n";
}
if ( ! -d $scbdir) {
  die "The EUROPLEXUS directory: $scbdir does not exist!\007\n";
}
if ( ! -d $mafdir) {
  system ("mkdir $mafdir");
}
if ( ! -d $libdir) {
  system ("mkdir $libdir");
}
if ( ! -d $moddir) {
  system ("mkdir $moddir");
}
if ( ! -d $exedir) {
  system ("mkdir $exedir");
}
#-------------------------------------------------------------------------------
# Read the number of the current evolution
#
$evonum = `cat $evonumfile`;
#chop $evonum;
$evonum =~ s/ //g;
print "===> Making version number  : \# $evonum\n";
#-------------------------------------------------------------------------------
# First, remove any previous made files
#
system("$del ../module/*.mod");
system("$del ../library/libplex.lib");
#system("$del ../library/libplex_c.lib");
system("$del ../library/libblas.lib");
system("$del ../manual_filtered/*.tex");
system("$del ../exe/europlexus.exe");
#-------------------------------------------------------------------------------
# Then, make sure ordo and epx_filter are built
#
print "===> Building ordo and epx_filter\n";
chdir "$utidir" || die "Can't chdir to $utidir!\007\n";
system ("df /fast ordo.f");
system ("df /fast epx_filter.f");
chdir "$makdir" || die "Can't chdir to $makdir!\007\n";
#-------------------------------------------------------------------------------
# Then, get and compile all sources
#
print "===> Building C library\n";
```

```
chdir "$sccdir" || die "Can't chdir to $sccdir!\007\n";
system ("make_all");
chdir "$makdir" || die "Can't chdir to $makdir!\007\n";
#
print "===> Building BLAS library\n";
chdir "$scbdir" || die "Can't chdir to $scbdir!\007\n";
system ("make_all");
chdir "$makdir" || die "Can't chdir to $makdir!\007\n";
#
print "===> Copying Fortran sources\n";
system ("cp -f ../source/*.ff .");
print "===> Compiling all Fortran sources\n";
system ("epx_cmp -q -o");
print "===> Storing modules\n";
system ("mv -f *.mod ../module");
print "===> Building library\n";
system ("lib /out:../library/libplex.lib *.obj");
print "===> Cleaning up\n";
system ("rm -f *.*");
#
print "===> Building executable\n";
system ("epx_get main");
system ("epx_cmp -o main");
system ("epx_lk -o");
print "===> Storing executable\n";
system ("mv -f epx.exe ../exe/europlexus.exe");
print "===> Cleaning up\n";
system ("rm -f *.*");
#-------------------------------------------------------------------------------
# Then, the benchmarks
#
print "===> Executing the benchmarks\n";
system ("epx_test_benchmarks -s");
print "===> Storing the benchmarks\n";
system ("mv -f *.listing ../bench");
system ("mv -f *.ps ../bench");
print "===> Cleaning up\n";
system ("rm -f *.*");
#-------------------------------------------------------------------------------
# Finally, the manuals
#
print "===> Copying manual sources\n";
system ("cp -f ../manual/*.ttx .");
print "===> Making manuals\n";
system ("epx_test_manuals");
print "===> Storing filtered manuals\n";
system ("mv -f *.tex ../manual_filtered");
print "===> Storing manuals\n";
system ("mv -f manual.dvi ../manual");
system ("mv -f manual.ps ../manual");
system ("mv -f manual.pdf ../manual");
system ("mv -f manual.html ../manual");
system ("mv -f manual_h.html ../manual");
system ("rm -f ../manual/hacha/*.*");
system ("rmdir ../manual/hacha");
system ("mkdir ../manual/hacha");
system ("mv -f hacha/*.* ../manual/hacha");
system ("rmdir hacha");
print "===> Cleaning up\n";
system ("rm -f *.*");
#
exit;


#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_make_evo.pl

```
#-------------------------------------------------------------------------------
# Creates an evo.txt file with the source, the benchmark, and the manual files
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$fiche = 0;                                # Fiche number
#-------------------------------------------------------------------------------
# Check number of arguments, must be 0 or 1
#
if ($#ARGV > 1) {
  die "Usage: epx_grep [-f fiche number] [-s] [-l]\007\n";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#       -f     gets the number of the fiche. Default value: 0.
#       -s     evolution sans fiche
#       -l     only lock file is generated
#
while ($ARGV[0] =~ /^-/) {
  $_ = shift;
  if (/^-\?(.*)/) {
     printf "Creates an evo.txt file with the source, the benchmark, and the man
ual files\n";
     printf "Process optional switches:\n";
     printf " -f nnn  Fiche number\n";
     printf " -s      Sans fiche\n";
     printf " -l      Generates only the lock file\n";
     exit;
  }
  elsif (/^-f$/) {
    $_ = shift;
    $fiche = $_;
  }
  elsif (/^-s(.*)/) {
    $fiche = -1;
  }
  elsif (/^-l(.*)/) {
    $lock = 1;
  }
  else {
    die "ERROR: unknown switch: $_\n";
  }
}
#
#system("ls *.ff *.epx *.ttx > evo_list.txt");
system("epx_diff -f *.ff > evo_list.txt");
system("epx_diff -b -f *.epx >> evo_list.txt");
system("epx_diff -m -f *.ttx >> evo_list.txt");
open (OFILE, "evo_list.txt");
while (<OFILE>) {
  chop;
  $fil = $_;
  if(index($_,"Command line switch")<0) {
    push @FileNames, $_;
  }
}
close OFILE;
print "Check epx_newer\n";
system("epx_newer");
system("epx_newer -b");
system("epx_newer -m");
if(!$lock){
  open (WRFILE, ">evo.txt");
}
open (LOCKFILE, ">lock.txt");
print "\nevo.txt contains the following files:\n";
foreach $filename (@FileNames) {
  print $filename."\n";
  print LOCKFILE $filename." old\n";
  if(!$lock){
    print WRFILE $filename;
    if($fiche == -1) {
       print WRFILE "\nEvolution sans fiche.\n\n";
    }
    else {
      printf WRFILE "\nEvolution associee avec la fiche de developpement #%04d",
$fiche.".\n\n";
      print WRFILE "\n\n";
    }
    print WRFILE "\n";
  }
}
if(!$lock){
  close WRFILE;
}
close LOCKFILE;
print "\n";
#
exit;
```

epx_manual.pl

```
#-------------------------------------------------------------------------------
# Open EUROPLEXUS manual
#-------------------------------------------------------------------------------
# Default values
#
$copy = "cp -p";
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  $errmsg = "The EUROPLEXUS environment variable is undefined!\007\n";
  die "$errmsg";
}
#-------------------------------------------------------------------------------
#
# Copy the manual to the %TEMP% directory
#
# Check that the temporary directory exists
$temp = $ENV{'TEMP'};
if ( ! -d $temp) {
  $errmsg = "The EUROPLEXUS temporary directory: $temp does not exist!\007\n";
  die "$errmsg";
}
#
$stdman = "$epx\\manual\\manual.pdf";
$tempman = "$temp\\manual.pdf";
system("$copy $stdman $tempman");
#system("$tempman \&");
system("start $tempman %*"); # Use start to free the console window ...
exit;

#-------------------------------------------------------------------------------
```

epx_mkbatch.pl

```
#-------------------------------------------------------------------------------
# Produce local EUROPLEXUS executable suitable for batch execution
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$out = "epxb.exe";
$errfil = "epx_mkbatch.err";
#-------------------------------------------------------------------------------
# Check number of arguments, must be 0
#
if ($#ARGV != -1) {                        # Index of last argument must be = -1
  $errmsg = "Usage: epx_mkbatch\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Retrieve ifwin.ff from sources if not existing already in the current dir
```

```
#
$ifwin="ifwin.ff";
if ( ! -r $ifwin ) {
  print "Retrieving ifwin.ff from sources!\n";
  #
  if ( system ("epx_get ifwin")) {
    $errmsg = "Unable to retrieve ifwin.ff from sources!\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
  #
  $remifw = "rm -f ifwin.*";
  print "Define remifw: $remifw \n";
}
#-------------------------------------------------------------------------------
# Compile ifwin.ff WITHOUT giving the QWIN filter keyword
#
print "Compiling ifwin.ff without the QWIN keyword\n";
#
if ( system ("epx_cmp -w -x -k WIN ifwin")) {
  $errmsg = "Unable to compile ifwin.ff!\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Link and produce the "batch" executable
#
if (system("epx_lk -o -f -e $out")) {
  #
  # Clean up ifwin.* if necessary
  #
  if (defined ($remifw)) {
    print ("Deleting (1) ifwin.*!\n");
    system("$remifw");
  }
  $errmsg = "Link ERROR(s)!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Remove ifwin.* (but only if it was previously extracted from sources)
#
if (defined ($remifw)) {
  print ("Deleting (2) ifwin.*!\n");
  system("$remifw");
}
#
print "The local BATCH executable: $out has been produced.\n";
#
exit;


#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_modrec.pl

```
#-------------------------------------------------------------------------------
# (recursive!) find list of files directly using a module file
#-------------------------------------------------------------------------------
# The argument is the (base) name of the module
#
$name = $ARGV[0];
#-------------------------------------------------------------------------------
# Set up name
#
$base = $name;
$base =~ s/\.ff$//;                         # Remove '.ff' extension if present
$_ = $base;
#
$rout = "_$base.txt";
#-------------------------------------------------------------------------------
# Avoid repeatedly building up the same file!
#
if (-r $rout ) {
  exit;
}
#-------------------------------------------------------------------------------
# Default values
#
$gnudir = $ENV{'GNUDIR'};
if ( ! defined $gnudir) {
  die "The GNUDIR environment variable is undefined!\007\n";
}
$sort = "$gnudir\\sort.exe"; # Full path needed! Else uses MS-DOS's sort!
$del  = "rm -f";
#-------------------------------------------------------------------------------
$temp = "_$base_temp.txt";
#-------------------------------------------------------------------------------
# Build list of direct users (source files that directly USE the module file)
#
system ("$del $rout");
#
print "Building the direct users list for module $base : ";
#
# Naive search would be:
# system ("epx_grep -l \"USE $base\" >>$rout");
#
# Better search is:
# Match zero or more white space, followed by USE, followed by one blank
# and zero or more white space, followed by module name, followed by
# either end-of-line, or a blank, or an exclamation point.
# The grep command is:
#   grep -l "^[[:space:]]*use [[:space:]]*$base\($\|[ !]\)"  ...
# but special characters must be escaped to pass through PERL
#
$com = "epx_grep -l \"\^\[\[:space:\]\]\*use \[\[:space:\]\]\*$base\\\(\$\\\|\[
\!\]\\\)\"";
#print "Command : $com\n";
system ("$com >>$rout");
#-------------------------------------------------------------------------------
# Sort list of direct users by eliminating multiple name occurrences
# (remove also full path names of source files, leaving only the file name)
#
system ("$sort -u $rout >$temp");
system ("$del $rout");
#
$epx = $ENV{'EUROPLEXUS'};
$epx1 = $epx;
$epx1 =~ s/\\/\\\\/;                        # Replace any \ by \\
#
open (INPUT, $temp);
open (OUTPUT, ">> $rout");
while (<INPUT>) {
  s/^$epx1\\source\\//;
  print (OUTPUT);
}
close OUTPUT;
close INPUT;
#
system ("$del $temp");
#
$num = `cat $rout | wc -l`;
chop $num;
$num =~ s/ //g;
#
print "$num files\n";
#-------------------------------------------------------------------------------
# do recursive module search
open (INPUT, $rout);
while (<INPUT>) {
  if ( m/^m_/ ) {
    $base = $_;
    $base =~ s/\.ff$//;                     # Remove '.ff' extension if present
    #
    # recursive call
    #
    system ("epx_modrec $base");
  }
}
close INPUT;
#-------------------------------------------------------------------------------
exit;
```

epx_modtree.pl

```
#-------------------------------------------------------------------------------
# Find list of files depending from (i.e. using directly or not) a module file
#-------------------------------------------------------------------------------
# Default values
#
$out  = "_users.txt";
$del  = "rm -f";
$errfil = "epx_modtree.err";
#-------------------------------------------------------------------------------
$gnudir = $ENV{'GNUDIR'};
if ( ! defined $gnudir) {
  $errmsg = "The GNUDIR environment variable is undefined!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
$sort = "$gnudir\\sort.exe"; # Full path needed! Else uses MS-DOS's sort!
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 1
#
if ($#ARGV != 0) {                          # Index of last argument must be = 0
  $errmsg = "Usage: epx_modtree name\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# The argument is the (base) name of the module
#
$name = $ARGV[0];
#-------------------------------------------------------------------------------
# Set up name
#
$base = $name;
$base =~ s/\.ff$//;                         # Remove '.ff' extension if present
$base0 = $base;
$_ = $base;
#
# Check that name starts by "m_"
#
if ( ! m/^m_/ ) {
  $errmsg = "Base name $base does not start by m_!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#
$rout = "_$base.txt";
$temp = "_$base_temp.txt";
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  $errmsg = "The EUROPLEXUS environment variable is undefined!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  $errmsg = "The EUROPLEXUS directory: $epx does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Cleanup any results of a previous search
#
system ("$del $out");
system ("$del _m_*.txt");
#-------------------------------------------------------------------------------
# Build list of direct users (source files that directly USE the module file)
#
system ("$del $rout");
#
print "Building the direct users list for module $base : ";
#
# Naive search would be:
# system ("epx_grep -l \"USE $base\" >>$rout");
#
```

```
# Better search is:
# Match zero or more white space, followed by USE, followed by one blank
# and zero or more white space, followed by module name, followed by
# either end-of-line, or a blank, or an exclamation point.
# The grep command is:
#   grep -l "^[[:space:]]*use [[:space:]]*$base\($\|[ !]\)"  ...
# but special characters must be escaped to pass through PERL
#
$com = "epx_grep -l \"\^\[\[:space:\]\]\*use \[\[:space:\]\]\*$base\\\(\$\\\|\[
\!\]\\\)\"";
#print "Command : $com\n";
system ("$com >>$rout");
#-------------------------------------------------------------------------------
# Sort list of direct users by eliminating multiple name occurrences
# (remove also full path names of source files, leaving only the file name)
#
system ("$sort -u $rout >$temp");
system ("$del $rout");
#
$epx1 = $epx;
$epx1 =~ s/\\/\\\\/;                          # Replace any \ by \\
#
open (INPUT, $temp);
open (OUTPUT, ">> $rout");
while (<INPUT>) {
  s/^$epx1\\source\\//;
  print (OUTPUT);
}
close OUTPUT;
close INPUT;
#
system ("$del $temp");
#
$num = `cat $rout | wc -l`;
chop $num;
$num =~ s/ //g;
#
print "$num files\n";
#-------------------------------------------------------------------------------
# do recursive module search
open (INPUT, $rout);
while (<INPUT>) {
  if ( m/^m_/ ) {
    $base = $_;
    $base =~ s/\.ff$//;                       # Remove '.ff' extension if present
    #
    # recursive call
    #
    system ("epx_modrec $base");
  }
}
close INPUT;
#-------------------------------------------------------------------------------
# Build list of users (direct and indirect), then sort it
#
system ("cat _m\*.txt > $temp");
system ("$sort -u $temp >$out");
system ("$del $temp");
#
$num = `cat $out | wc -l`;
chop $num;
$num =~ s/ //g;
#
print "\nALL users of $base0 are in $out : $num files\n";
#-------------------------------------------------------------------------------
exit;


#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_newer.pl

```
#-------------------------------------------------------------------------------
# Verify whether there are newer versions of source files
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$ext = "ff";                              # By default, source file
$dir = "source";                          # By default, source directory
#-------------------------------------------------------------------------------
# Check number of arguments, must be 0 or 1
#
if ($#ARGV < -1) {                        # Index of last argument must be >= -
1
  die "Usage: epx_newer [-b] [-m] [-v] [-?] \007\n";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#       -b     benchmark, i.e. ".epx" file, not source (.ff) file
#       -m     manual, i.e. ".ttx" file, not source (.ff) file
#       -v     validation, i.e. ".ttx" file, not source (.ff) file
#
while ($ARGV[0] =~ /^-/) {
  $_ = shift;
  if (/^-\?(.*)/) {
    printf "Verify whether there are newer versions of source files\n";
    printf "-b     benchmark, i.e. '.epx' file, not source (.ff) file\n";
    printf "-m     manual, i.e. '.ttx' file, not source (.ff) file\n";
    printf "-v     validation, i.e. '.ttx' file, not source (.ff) file\n";
    exit;
 }
  elsif (/^-b(.*)/) {
    $ext = "epx";
    $dir = "bench";
    printf "Command line switch: benchmark file\n";
  }
  elsif (/^-m(.*)/) {
    $ext = "ttx";
    $dir = "manual";
    printf "Command line switch: manual file\n";
  }
  elsif (/^-v(.*)/) {
    $ext = "vld";
    $dir = "validate";
    printf "Command line switch: validation file\n";
  }
  else {
    die "ERROR: unknown switch: $_\n";
  }
}
#-------------------------------------------------------------------------------
# Check if the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  die "The EUROPLEXUS environment variable is undefined!\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  die "The EUROPLEXUS directory: $epx does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Build up list of local source file names
#
@FileNames = glob("*.$ext");
#-------------------------------------------------------------------------------
# Loop on file(s) to be compared
#
foreach $file (@FileNames) {
#-------------------------------------------------------------------------------
# Verify that the file exists in the appropriate directory
#
    $from = "$epx\\$dir\\$file";                # Full file name
    if ( ! -r $from ) {
      print "File $file does not exist in $epx\\$dir!\007\n";
    }
    else {
#-------------------------------------------------------------------------------
# Verify that the file exists in the local directory
#
      if ( ! -r $file ) {
        die "File $file does not exist in local directory!\007\n";
      }
#-------------------------------------------------------------------------------
# Diff file header
#
#     system("$cmd $opt $from $file $more");
      $fromhead = `head -1 $from`;
      $filehead = `head -1 $file`;
#     print("$fromhead");
#     print("$filehead\n");
      if ( $fromhead eq $filehead ) {
        print("OK $file\n");
      }
      else {
        print("\nNewer $file !!!\007\n");
        print("$fromhead");
        print("$filehead\n");
      }
#
    }
#
}
exit;
```

epx_obsolete.pl

```
#-------------------------------------------------------------------------------
# Set aside obsolete file(s)
#-------------------------------------------------------------------------------
# Default values
#
$del = "rm -f";
$move = "mv -f";
$ext = "ff";                              # By default, source file
$dir = "source";                          # By default, source directory
$sour = 1;                                # By default, it is a source file
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 1or 2
#
if ($#ARGV > 1) {                       # Index of last argument must be = 0 or 1
  die "Usage: epx_obsolete [-i] [-b] [-m] [-v] name[.<extension>]\007\n";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#       -i     include file, i.e. ".inc" file, not source (.ff) file
#       -b     benchmark, i.e. ".epx" file, not source (.ff) file
#       -m     manual, i.e. ".ttx" file, not source (.ff) file
#       -v     validation, i.e. ".vld" file, not source (.ff) file
#
while ($ARGV[0] =~ /^-/) {
  $_ = shift;
  if (/^-i(.*)/) {
    $ext = "inc";
    $dir = "include";
    undef $sour;
    printf "Command line switch: include file\n";
  }
  elsif (/^-b(.*)/) {
    $ext = "epx";
    $dir = "bench";
    undef $sour;
    $ben = 1;
    printf "Command line switch: bench file\n";
  }
  elsif (/^-m(.*)/) {
    $ext = "ttx";
    $dir = "manual";
```

```
    $manfil = "yes";
    undef $sour;
    printf "Command line switch: manual file\n";
  }
  elsif (/^-v(.*)/) {
    $ext = "vld";
    $dir = "validate";
    undef $sour;
    $val = 1;
    printf "Command line switch: validation file\n";
  }
  else {
    die "ERROR: unknown switch: $_\n";
  }
}
#------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  die "The EUROPLEXUS environment variable is undefined!\007\n";
}
#------------------------------------------------------------------------------
# Check that the EUROPLEXUS directories exist
#
if ( ! -d $epx ) {
  die "The EUROPLEXUS directory: $epx does not exist!\007\n";
}
#
$hisdir = "$epx\\history";
if ( ! -d $hisdir ) {
  die "The history directory: $hisdir does not exist!\007\n";
}
#
$hisobs = "$hisdir\\obsolete";
if ( ! -d $hisobs ) {
  die "The history obsolete directory: $hisobs does not exist!\007\n";
}
#
$moddir = "$epx\\module";
if ( ! -d $moddir ) {
  die "The module directory: $moddir does not exist!\007\n";
}
#
$objlib = "$epx\\library\\libplex.lib";
if ( ! -r $objlib ) {
  die "The object library: $objlib does not exist!\007\n";
}
#
$manfildir = "$epx\\manual_filtered";
if ( ! -d $manfildir ) {
  die "The filtered manual directory: $manfildir does not exist!\007\n";
}
#
$valdir = "$epx\\validate";
if ( ! -d $valdir ) {
  die "The validation directory: $valdir does not exist!\007\n";
}
#------------------------------------------------------------------------------
# Set up file name
#
$name = $ARGV[0];                           # Get first and only argument
$base = $name;
$base =~ s/\.$ext$//;                       # Remove extension if present
$file = "$base.$ext";
#------------------------------------------------------------------------------
# Verify that the file exists in the corresponding directory
#
$fromdir = "$epx\\$dir";                    # Full directory name
$from = "$fromdir\\$file";                  # Full file name
if ( ! -r $from ) {
  die "File $from does not exist!\007\n";
}
#------------------------------------------------------------------------------
# Target directory and target file
#
$todir =  "$fromdir\\obsolete";            # Full directory name
if ( ! -d $todir ) {
  die "The target directory: $todir does not exist!\007\n";
}
$to = "$todir\\$file";                      # Full file name
if ( -r $to ) {
  printf "Removing $to\n";
  system("$del $to");
}
printf "Moving $from $to\n";
system("$move $from $to");
#------------------------------------------------------------------------------
# Additional treatment for bench files : move also auxiliary files
#
if ( defined $ben ) {
  $fromfiles = "$fromdir\\$base.\*";
  printf "Moving $fromfiles $todir\n";
  system("$move $fromfiles $todir");
}
#------------------------------------------------------------------------------
# Additional treatment for validation files : move also auxiliary files
#
if ( defined $val ) {
  $fromfiles = "$fromdir\\$base.\*";
  printf "Moving $fromfiles $todir\n";
  system("$move $fromfiles $todir");
}
#------------------------------------------------------------------------------
# Treat also corresponding history file, if any
#
$fromhis = "$hisdir\\$base.his";
if ( -r $fromhis ) {
  $tohis = "$hisobs\\$base.his";            # Full file name
  if ( -r $tohis ) {
    printf "Removing $tohis\n";
    system("$del $tohis");
  }
  printf "Moving $fromhis $tohis\n";
  system("$move $fromhis $tohis");
}
#------------------------------------------------------------------------------
# Additional treatment for source files : remove corresponding object file
# from the library, and also remove corresponding .mod file if any
#
if ( defined $sour ) {
  printf "Removing $base.obj from $objlib\n";
  system("LIB /REMOVE:$base.obj $objlib");
  $mod = "$moddir\\$base.mod";              # Full file name
  if ( -r $mod ) {
    printf "Removing $mod\n";
    system("$del $mod");
  }
}
#------------------------------------------------------------------------------
# Additional treatment for manual files : remove corresponding filtered file
# from the manual_filtered directory
#
if ( defined $manfil ) {
  $tex = "$manfildir\\$base.tex";           # Full file name
  if ( -r $tex ) {
    printf "Removing $tex\n";
    system("$del $tex");
  }
}
#------------------------------------------------------------------------------
exit;
```

epx_ordo.pl

```
# Create a file epx_ordo.txt containing the list of all f90 files *.ff
# in the current directory **in the order in which they must be compiled**
#------------------------------------------------------------------------------
# Check number of arguments, must be = 0
#
if ($#ARGV != -1) {                         # Index of last argument must be = -1
  die "Usage: epx_ordo\007\n";
}
#------------------------------------------------------------------------------
$tmp_yes = "_tmp_yes.txt";  # modules with use (contain at least one use stmt)
$tmp_occ = "_tmp_occ.txt";  # use stmts in modules
$tmpfile = "_tmp_ordo.txt"; # n1, n2; modules with use; use stmts in modules
$outfile = "epx_ordo.txt";  # sources listed in order of compilation
$del = "rm -f";
#------------------------------------------------------------------------------
# Delete the output file and temporary file if already existing
#
if ( -r $outfile ) {
  print "Removing old file $outfile\n";
  system ("$del $outfile");
}
system ("touch $outfile");
if ( -r $tmpfile ) {
  print "Removing old file $tmpfile\n";
  system ("$del $tmpfile");
}
#------------------------------------------------------------------------------
# Count the *.ff files
#
$ntot = `ls *.ff | wc -l`;       # number of .ff files in current directory
chop $ntot;
#
if ($ntot == 0) {
  die "There are no .ff files to treat!\007\n";
}
else {
  print "There are $ntot source files to treat\n";
}
#------------------------------------------------------------------------------
# Count the module (m_*.ff) files
#
$nmod = `ls m_*.ff | wc -l`;     # number of module files in current directory
chop $nmod;
print "of which  $nmod are module files\n";
if ($nmod > 0) {
#------------------------------------------------------------------------------
# STEP 1 : search module files m_*.ff which do **not** contain 'USE'
#          and write their list in $outfile
#
  $com_no = "grep -i -L \"\^\[\[:space:\]\]\*use \" m\_\*\.ff";
  #print "Command : $com_no\n";
  $n0 = `$com_no | wc -l`;     # number of module files not containing 'USE'
  chop $n0;
  print "n0: n. of modules not containing any USE statement      = $n0\n";
  system ("$com_no >> $outfile");
#------------------------------------------------------------------------------
# STEP 2 : treat module files m_*.ff which **do** contain 'USE'
#          and append their list to $outfile in the correct order
#
  $com_yes = "grep -i -l \"\^\[\[:space:\]\]\*use \" m\_\*\.ff";
  #print "Command : $com_yes\n";
  $n1 = `$com_yes | wc -l`;     # number of module files containing 'USE'
  chop $n1;
  print "n1: n. of modules containing at least one USE statement = $n1\n";
  system ("$com_yes > $tmp_yes");
#
  $com_occ = "grep -i    \"\^\[\[:space:\]\]\*use \" m\_\*\.ff";
  #print "Command : $com_occ\n";
  $n2 = `$com_occ | wc -l`;     # number of occurrences of 'USE' in module files
  chop $n2;
  print "n2: total number of USE statements in modules           = $n2\n";
  system ("$com_occ > $tmp_occ");
#
  if ($n1 > 0) {
    open (OUT, "> $tmpfile");
    print OUT "$n1 $n2\n";
    close OUT;
#
    system ("$com_yes >> $tmpfile");
#
    open (OUT, ">> $tmpfile");
    open (IN, $tmp_occ);
    while (<IN>) {
      #print "$_";
      chomp;            # a more sophisticated version of 'chop'
      s/:/  /;
      ($a, $b, $c) = split (" ");
      $d = lc ($c);     # converts to lower-case
      print OUT $a, "    ","$d.ff", "\n";
```

```
    }
    close IN;
    close OUT;
#
    system ("ordo.exe < $tmpfile >> $outfile");
  }
}
#-------------------------------------------------------------------------------
# STEP 3 : add to list the source files other than modules, if any
#
$com_oth = "grep -v \"\^m\_\"";
#print "Command : $com_oth\n";
$n3 = `ls *.ff | $com_oth | wc -l`; # number of non-module source files
chop $n3;
print "n3: total number of non-module source files             = $n3\n";
system ("ls *.ff | $com_oth >> $outfile");
#
# Make sure there are no trailing blanks and fix <LF> / <CR><LF> in $outfile
#
system ("detrail $outfile");
#-------------------------------------------------------------------------------
# clean up
#
#if ( -r $tmpfile ) {
#  print "Removing old file $tmpfile\n";
#  system ("$del $tmpfile");
#}
#if ( -r $tmp_yes ) {
#  print "Removing old file $tmp_yes\n";
#  system ("$del $tmp_yes");
#}
#if ( -r $tmp_occ ) {
#  print "Removing old file $tmp_occ\n";
#  system ("$del $tmp_occ");
#}
#-------------------------------------------------------------------------------
exit;
```

epx_Ordo_Fort.f

```
      PROGRAM ordo
      IMPLICIT NONE
      INTEGER ::
     >  nt,        ! n. of modules containing at least one USE statement
     >  nlie       ! total n. of USE statements in modules
      CHARACTER(26), ALLOCATABLE ::
     >  list(:),   ! modules containing at least one USE, in alpha order
     >  forwd(:),  ! modules containing USE
     >  bckwd(:)   ! USEd modules
      INTEGER, ALLOCATABLE ::
     >  class(:),  ! levels of modules containing at least one USE statement
     >  liais(:),  ! USE relationships
     >  kbckwd(:),
     >  kforwd(:)
      CHARACTER(26) ::
     >  b,
     >  f
      INTEGER ::
     >  i,
     >  j,
     >  iforwd,
     >  ibckwd,
     >  iliais,
     >  mxi,
     >  iaux,
     >  ier,
     >  lung
      INTEGER, EXTERNAL :: ilyest, norm_class
* read data
      READ *, nt, nlie
      ALLOCATE (list(nt))
      ALLOCATE (forwd(nlie))
      ALLOCATE (bckwd(nlie))
      ALLOCATE (class(nt))
      ALLOCATE (kforwd(nlie))
      ALLOCATE (kbckwd(nlie))
      ALLOCATE (liais(nlie))
      READ *, list
      READ *, (forwd(i), bckwd(i), i = 1, nlie)
* initializations
      class(:)  = 1
      kforwd(:) = 0
      kbckwd(:) = 0
      liais(:)  = 0
* on vire les liaisons dont le bckwd n'est pas dans la liste
      DO i = 1, nlie
        b = bckwd(i)
        iliais = ilyest (nt, list, b)
        liais(i) = iliais
        IF (iliais /= 0) THEN
          kbckwd(i) = iliais
          f = forwd(i)
          kforwd(i) = ilyest (nt, list, f)
        ENDIF
      END DO
*
      mxi = 1
      DO j = 1, nt
        iaux = norm_class (nt, class)
        DO i = 1, nlie
           IF (liais(i) == 0 ) CYCLE
           iforwd = kforwd(i)
           ibckwd = kbckwd(i)
           class(iforwd) = MAX (class(iforwd), class(ibckwd)+1)
           mxi = MAX (mxi, class(iforwd))
        END DO
*
* modifs du 01Aout2005:
* on teste si norm_class ne change pas => convergence
* alors que avant on testait sur la croissance de mxi
*
        ier = norm_class (nt, class)
        IF (ier == iaux) THEN
          ier = 0
          EXIT
        ENDIF
      END DO ! j = 1, nt
*
      IF (ier /= 0) THEN
* ier /= 0 : probleme
        WRITE (0, 1001) ier
        DO j = 1, nt
          IF (class(j) > nt) WRITE (0, 1100) list(j)
        END DO
        CALL report_cycle (nt, nlie, list, forwd, bckwd)
        STOP 'ORDO ERROR : IER /= 0'
      ENDIF
*
      IF (mxi > nt) THEN
* le nbre de classes > nt : probleme
        WRITE (0, 1000) mxi, nt
        DO j = 1, nt
          IF (class(j) > nt) WRITE (0, 1100) list(j)
        END DO
        CALL report_cycle (nt, nlie, list, forwd, bckwd)
        STOP 'ORDO ERROR : MXI > NT'
      ENDIF
*
      DO i = 1, mxi
        DO j = 1, nt
          IF (class(j) == i) THEN
!!            lung = INDEX (list(j), '.ff') + 2
!!            PRINT 2000, list(j)(1:lung)
            PRINT 2000, list(j)
          ENDIF
        END DO
      END DO
*
      STOP 'ORDO : NORMAL END'
*
 1000 FORMAT ('Cyclage: Nbre de classes = ',I4,
     >        ' > nbre de fichiers = ',I4)
 1001 FORMAT ('Cyclage: ier = ',I10)
 1100 FORMAT ('fichiers concernés : ',A)
 2000 FORMAT (A)
*
      END PROGRAM ordo
*=======================================================================
      MODULE m_listused
      IMPLICIT NONE
      PUBLIC :: listused
      TYPE listused
        INTEGER :: nusato
        INTEGER, POINTER :: usato(:)
      END TYPE listused
      TYPE(listused), POINTER :: listu(:)
      END MODULE m_listused
*=======================================================================
      INTEGER FUNCTION ilyest (nt, list, dad)
*
* returns j : > 0  if dad is j-th element of list(:)
*             == 0 if dad is not contained in list(:)
*
      IMPLICIT NONE
* args
      INTEGER, INTENT(IN) :: nt
      CHARACTER(26), INTENT(IN) :: list(nt)
      CHARACTER(26), INTENT(IN) :: dad
*
* locals
      INTEGER :: j
*
      ilyest = 0
      DO j = 1, nt
        IF (dad == list(j)) THEN
          ilyest = j
          EXIT
        ENDIF
      END DO
*
      END FUNCTION ilyest
*=======================================================================
      INTEGER FUNCTION ilyest_int (nt, list, dad)
*
* returns j : > 0  if dad is j-th element of list(:)
*             == 0 if dad is not contained in list(:)
*
      IMPLICIT NONE
* args
      INTEGER, INTENT(IN) :: nt
      INTEGER, INTENT(IN) :: list(nt)
      INTEGER, INTENT(IN) :: dad
*
* locals
      INTEGER :: j
*
      ilyest_int = 0
      DO j = 1, nt
        IF (dad == list(j)) THEN
          ilyest_int = j
          EXIT
        ENDIF
      END DO
*
      END FUNCTION ilyest_int
*=======================================================================
      INTEGER FUNCTION norm_class (nt, class)
*
* returns the sum of the nt items in class(:)
*
      IMPLICIT NONE
*
      INTEGER, INTENT(IN) :: nt
      INTEGER, INTENT(IN) :: class(nt)
*
* locals
      INTEGER :: ip
*
      norm_class = 0
      DO ip = 1, nt
        norm_class = norm_class + class(ip)
      END DO
*
```

```
      END FUNCTION norm_class
*=====================================================================
      SUBROUTINE report_cycle (nmod, nuse, modul, mouse, used)
*
* detect cycle and report it
*
* there are nmod modules each containing at least one USE statement
*
* their names are contained (in alpha order, but this is irrelevant)
* into modul(:)
*
* there are nuse occurrences of the USE statement altogether in these
* nmod modules (therefore it must be nuse >= nmod)
*
* for each USE statement, mouse is the module, used is the used module
* i.e. module mouse contains "USE used".
*
      USE m_listused
*
      IMPLICIT NONE
*
* args
      INTEGER, INTENT(IN) :: nmod, nuse
      CHARACTER(26), INTENT(IN) :: modul(nmod), mouse(nuse), used(nuse)
*
* locals
      INTEGER :: i, m, u, nold, ncycl
      CHARACTER(26) :: mou, usd
      LOGICAL :: lcycle
      INTEGER, POINTER :: cycl(:), done(:)
*
      INTEGER, EXTERNAL :: ilyest
*
      NULLIFY (listu)
      NULLIFY (cycl)
      NULLIFY (done)
*
      ALLOCATE (listu(nmod))
      DO i = 1, nmod
        listu(i)%nusato = 0
        NULLIFY (listu(i)%usato)
      END DO
*
      DO i = 1, nuse
        mou = mouse(i)
        m = ilyest (nmod, modul, mou)
        IF (m == 0) STOP 'REPORT_CYCLE M = 0'
        usd = used(i)
        u = ilyest (nmod, modul, usd)
        IF (u > 0) THEN
          listu(m)%nusato = listu(m)%nusato + 1
        ENDIF
      END DO
*
      DO i = 1, nmod
        IF (listu(i)%nusato > 0) THEN
          ALLOCATE (listu(i)%usato(listu(i)%nusato))
        ENDIF
        listu(i)%nusato = 0
      END DO
*
      DO i = 1, nuse
        mou = mouse(i)
        m = ilyest (nmod, modul, mou)
        IF (m == 0) STOP 'REPORT_CYCLE M = 0'
        usd = used(i)
        u = ilyest (nmod, modul, usd)
        IF (u > 0) THEN
          listu(m)%nusato = listu(m)%nusato + 1
          listu(m)%usato(listu(m)%nusato) = u
        ENDIF
      END DO
*
      DO i = 1, nmod
        IF (listu(i)%nusato > 0) THEN
          nold = listu(i)%nusato
          CALL iordo (listu(i)%nusato, listu(i)%usato, 1) ! vire doubles
          listu(i)%usato(listu(i)%nusato+1:nold) = 0
        ENDIF
      END DO
*
      ALLOCATE (cycl(nmod))
      ALLOCATE (done(nmod))
      done(:) = 0
      DO i = 1, nmod - 1
        IF (done(i) == 1) CYCLE
        IF (listu(i)%nusato == 0) CYCLE
        ncycl = 0
        cycl(:) = 0
        lcycle = .FALSE.
        CALL rec_use (i, nmod, modul, cycl, ncycl, lcycle, done)
        done(i) = 1
        IF (lcycle) EXIT
      END DO
      DEALLOCATE (cycl)
      DEALLOCATE (done)
*
      DEALLOCATE (listu)
*
      END SUBROUTINE report_cycle
*=====================================================================
      RECURSIVE SUBROUTINE rec_use (i, nmod, modul, cycl, ncycl, lcycle,
     >                              done)
*
      USE m_listused
*
      IMPLICIT NONE
*
* args
      INTEGER, INTENT(IN) :: i, nmod
      CHARACTER(26), INTENT(IN) :: modul(nmod)
      INTEGER, INTENT(INOUT) :: ncycl, cycl(nmod), done(nmod)
      LOGICAL, INTENT(OUT) :: lcycle
*
* locals
      INTEGER :: k, p, m, n, rcycl
*
      INTEGER, EXTERNAL :: ilyest_int
*
      IF (listu(i)%nusato == 0) RETURN
      ncycl = ncycl + 1
      cycl(ncycl) = i
*
      DO k = 1, listu(i)%nusato
        n = listu(i)%usato(k)
        p = ilyest_int (ncycl-1, cycl, n)
        IF (p > 0) THEN
          ncycl = ncycl - p + 1
          DO m = 1, ncycl
            cycl(m) = cycl(m+p-1)
          END DO
          cycl(ncycl+1:nmod) = 0
          lcycle = .TRUE.
          CALL print_cycle (ncycl, cycl, nmod, modul)
          RETURN
        ENDIF
        rcycl = ncycl
        CALL rec_use (n, nmod, modul, cycl, rcycl, lcycle, done)
        done(n) = 1
        IF (lcycle) RETURN
      END DO
*
      END SUBROUTINE rec_use
*=====================================================================
      SUBROUTINE print_cycle (ncycl, cycl, nmod, modul)
*
      IMPLICIT NONE
*
* args
      INTEGER, INTENT(IN) :: ncycl, cycl(ncycl), nmod
      CHARACTER(26), INTENT(IN) :: modul(nmod)
*
* locals
      INTEGER :: i, ip1
*
      WRITE(0,1000) ncycl
 1000 FORMAT('Cycle detected in following chain of', I4, ' modules:')
      DO i = 1, ncycl
        IF (i < ncycl) THEN
          ip1 = i + 1
        ELSE
          ip1 = 1
        ENDIF
        WRITE(0,1001) cycl(i), modul(cycl(i)), ' USE ', cycl(ip1),
     >                modul(cycl(ip1))
 1001   FORMAT(I4,2X,A26,A5,I4,2X,A26)
      END DO
*
      END SUBROUTINE print_cycle



*=====================================================================
C IORDO                       TOUS HBUNG     06/03/11 20:06:16
!
!---------------- algorithme de TRI ---------------------------
!  Feb 2006
!     Avant on utilise pour le tri, l'algorithme de HOARE ecrit
!     en Fortran.
!
!     Ce algorithme est tres performant et le temps CPU ~ N*log(N)
!     Mais lorsque le tableau initial est DEJA ORDONNE
!     on a CPU ~ N*N (Cf. Numerical recipes in Fortran 77 : The Art
!     of scientific Computing (1992) Cambridge University Press)
!
!     On s'apercoit de ce grave inconvenient dans LIRLEC, avec le
!     maillage de grande taille (nelem > 1E6), lorsque on tente de
!     trier la liste dans l'ordre croissant
!     (ex pour 1 tableau de 1200000, iordo prend 1700 s CPU)
!
!     Nous cherchons un autre algorithme de tri, il y a l'algo de
!     HOARE, modifié par SINGLETON. Le Fortran est ecrit par JONES &
!     WISNIEWSKI  qui est performant quelque soit la configuration
!     initiale. Nousl'avons appelé IORDO_HSJW (les 4 initiales
!     Hoare, Singleton, Jones, Wisniewski).
!     Nous avons testé (sur Pentium 2GHz)

!     1) Tableau initial DEJA ORDONNE
!       Longueur du tableau     TCPU_HOARE(s)        TCPU_HSJW(s)
!            120000                 8.0                  0.00
!            570000               334.2                  0.05
!           1230000              1713.8                  0.11
!     2) Tableau initial ALEATOIRE
!       Longueur du tableau     TCPU_HOARE(s)        TCPU_HSJW(s)
!            120000                 0.02                 0.0
!            570000                 0.12                 0.14
!           1230000                 0.29                 0.33
!
!     Ce tableau montre que l'algorithme de Hoare est tres performant
!     mais le temps CPU croit fortement lors que le tableau est DEJA
!     ORDONNE.
!     Quant a L'algorithme HOARE-SINGLETON, il reste toujours ~ N*log(N),
!     quelque soit la nature du tableau initial.
!
!     Comme dans la plupart des cas dans Europlexus, le tableau initial
!     est DEJA ORDONNE (par exemple la lecture des listes d'elements ou
!     de noeud), la routine standard IORDO fait maintenant appel a
!     IORDO_HSJW.
!
!    Routines cotenues dans ce fichier :
!     1) IORDO_HSJW:  algorithme de HOARE modifie par SINGLRTON
!     1) IORDO_HOARE: algorithme de HOARE initial
!     2) IORDO_BUCKET:  utilise le tri par casier
!                                                    H. Bung
!
      SUBROUTINE IORDO(N,ITAB,IOP)
*
*     ----------------------------------------------------------------
*
*       tri rapide utilisant qsort dans l'ordre croissant ou decroissant
*                                              h.  bung 04-88
*     ----------------------------------------------------------------
*
*             n        : longueur de itab
*             itab     : tableau a modifier
*          croissant
```

```
*             iop      : =0  les valeurs identiques sont consecutives
*                        =1  on vire les doubles ( n est modifie )
*          decroissant
*             iop      : =-2 les valeurs identiques sont consecutives
*                        =-1 on vire les doubles ( n est modifie )
*             si n .le. 1     : retour sans modification
*
*
*
*
      IMPLICIT NONE
*
*
      INTEGER, INTENT(INOUT) :: N,ITAB(*)
      INTEGER, INTENT(IN)    :: IOP
*
      INTEGER :: I, NN
*
*
*
      IF(N < 2) RETURN

      IF(IOP < 0) THEN
!---      tri decroissant: phase prelim. itab(i) = -itab(I)
         DO I=1,N
          ITAB(I) = - ITAB(I)
         END DO
      ENDIF

!!-----   Tri rapide : HOARE-SINGLETON
      CALL IORDO_HSJW(N, ITAB)
*
*
*-----   on elimine les doubles (si abs(iop)=1 )
      IF(IABS(IOP) == 1) THEN
         NN=1
         DO I=2,N
           IF(ITAB(I) == ITAB(NN)) CYCLE
           NN=NN+1
           IF(NN.NE.I) ITAB(NN)=ITAB(I)
         END DO
!-----       N est modifie
         N=NN
      ENDIF

      IF(IOP < 0) THEN
!---      tri decroissant: on remet itab(i) = -itab(I)
         DO I=1,N
          ITAB(I) = - ITAB(I)
         END DO
      ENDIF
*
      END SUBROUTINE IORDO

*=====================================================================

      SUBROUTINE IORDO_HSJW(N, IX)

!!  Tri Hoare-Singleton, Fortran ecrit par Jones & Wisniewski (SSORT)

****begin prologue  ssort
****date written   761101   (yymmdd)
****revision date  820801   (yymmdd)
****category no.  n6a2b1
****keywords  quicksort,singleton quicksort,sort,sorting
****author  jones, r. e., (snla)
*           wisniewski, j. a., (snla)
****purpose  ssort sorts array x and optionally makes the same
*            interchanges in array y.  the array x may be sorted in
*            increasing order or decreasing order.  a slightly modified
*            quicksort algorithm is used.
****description
*
*     written by rondall e. jones
*     modified by john a. wisniewski to use the singleton quicksort
*     algorithm.  date 18 november 1976.
*
*     abstract
*         ssort sorts array x and optionally makes the same
*         interchanges in array y.  the array x may be sorted in
*         increasing order or decreasing order.  a slightly modified
*         quicksort algorithm is used.
*
*     reference
*         singleton, r. c., algorithm 347, an efficient algorithm for
*         sorting with minimal storage, cacm,12(3),1969,185-7.
*
*     description of parameters
*         x - array of values to be sorted   (usually abscissas)
*         y - array to be (optionally) carried along
*         n - number of values in array x to be sorted
*         kflag - control parameter
*              =2  means sort x in increasing order and carry y along.
*              =1  means sort x in increasing order (ignoring y)
*              =-1 means sort x in decreasing order (ignoring y)
*              =-2 means sort x in decreasing order and carry y along.
****references  singleton,r.c., algorithm 347, an efficient algorithm
*                 for sorting with minimal storage, cacm,12(3),1969,
*                 185-7.
****routines called  xerror
****end prologue  ssort

      IMPLICIT NONE

!---      Dummy variables
      INTEGER, INTENT(IN) :: N
      INTEGER, INTENT(INOUT) :: IX(N)

!---       local variables
!!     M=DIM(IL)=DIM(IU)) : on peur traiter jusqu'a N < 2**M
      INTEGER :: IL(33),IU(33)
!!hb      INTEGER :: KFLAG=1     !  ordre croissant
      INTEGER :: NN, M, I, J, K, L, IJ
      REAL    ::  R
      INTEGER :: IT,KT


      NN = N
*
* sort x only
*
  100 CONTINUE
      M=1
      I=1
      J=NN
      R=.375
  110 IF (I .EQ. J) GO TO 155
  115 IF (R .GT. .5898437) GO TO 120
      R=R+3.90625E-2
      GO TO 125
  120 R=R-.21875
  125 K=I
*                                 select a central element of the
*                                 array and save it in location it
      IJ = I + (J-I) * R
      IT=IX(IJ)
*                                 if first element of array is greater
*                                 than it, interchange with it
      IF (IX(I) .LE. IT) GO TO 130
      IX(IJ)=IX(I)
      IX(I)=IT
      IT=IX(IJ)
  130 L=J
*                                 if last element of array is less than
*                                 it, interchange with it
      IF (IX(J) .GE. IT) GO TO 140
      IX(IJ)=IX(J)
      IX(J)=IT
      IT=IX(IJ)
*                                 if first element of array is greater
*                                 than it, interchange with it
      IF (IX(I) .LE. IT) GO TO 140
      IX(IJ)=IX(I)
      IX(I)=IT
      IT=IX(IJ)
      GO TO 140
  135 KT=IX(L)
      IX(L)=IX(K)
      IX(K)=KT
*                                 find an element in the second half of
*                                 the array which is smaller than it
  140 L=L-1
      IF (IX(L) .GT. IT) GO TO 140
*                                 find an element in the first half of
*                                 the array which is greater than it
  145 K=K+1
      IF (IX(K) .LT. IT) GO TO 145
*                                 interchange these elements
      IF (K .LE. L) GO TO 135
*                                 save upper and lower subscripts of
*                                 the array yet to be sorted
      IF (L-I .LE. J-K) GO TO 150
      IL(M)=I
      IU(M)=L
      I=K
      M=M+1
      GO TO 160
  150 IL(M)=K
      IU(M)=J
      J=L
      M=M+1
      GO TO 160
*                                 begin again on another portion of
*                                 the unsorted array
  155 M=M-1
      IF (M .EQ. 0) GO TO 300
      I=IL(M)
      J=IU(M)
  160 IF (J-I .GE. 1) GO TO 125
      IF (I .EQ. 1) GO TO 110
      I=I-1
  165 I=I+1
      IF (I .EQ. J) GO TO 155
      IT=IX(I+1)
      IF (IX(I) .LE. IT) GO TO 165
      K=I
  170 IX(K+1)=IX(K)
      K=K-1
      IF (IT .LT. IX(K)) GO TO 170
      IX(K+1)=IT
      GO TO 165
*
  300 CONTINUE

      END SUBROUTINE IORDO_HSJW

*=====================================================================

      SUBROUTINE IORDO_HOARE(N,ITAB,IOP)
*
*     ----------------------------------------------------------------
*
*       tri rapide (hoare) ordre croissant ou decroissant
*                                                 h.  bung 04-88
*     ----------------------------------------------------------------
*
*             n        : longueur de itab
*             itab     : tableau a modifier
*          croissant
*             iop      : =0  les valeurs identiques sont consecutives
*                        =1  on vire les doubles ( n est modifie )
*          decroissant
*             iop      : =-2 les valeurs identiques sont consecutives
*                        =-1 on vire les doubles ( n est modifie )
*             si n .le. 1     : retour sans modification
*
*
*
*
      IMPLICIT NONE
*
*
      INTEGER, INTENT(INOUT) :: N,ITAB(*)
      INTEGER, INTENT(IN)    :: IOP
*
      INTEGER :: I,NN,IR,IRA,IAUX,IRB,IQ,IA,IB,J,LPIVOT
```

```
*
*-----    variables locales pour les piles de doublets
*
      INTEGER, PARAMETER :: LGPILE=100
      INTEGER :: KOPF, IPIL1(LGPILE),IPIL2(LGPILE)
*
*
      IF(N < 2) RETURN
*
      IPIL1(1)=1        !AVANT: CALL PILINI & CALL PILALL(1,N))
      IPIL2(1)=N
      KOPF=1

      I=0
      J=0
*
  1   CONTINUE
!!!avant :       CALL PILFRE(IA,IB)
      IA=IPIL1(KOPF)
      IB=IPIL2(KOPF)
      KOPF=KOPF-1
  2   IF(IA.LT.IB) THEN
         LPIVOT=ITAB(IA)
*--          exploration de ia+1 a ib
         IQ=IA+1
         IR=IB
   3     CONTINUE
         IF(IQ.LE.IR) THEN
             J=IR-1
             DO I=IQ,J
                IF(ITAB(I).GT.LPIVOT) GOTO 5
             END DO
   5         IQ=I+1
             DO J=IR,I+1,-1
                IF(ITAB(J).LT.LPIVOT) GOTO 7
             END DO
   7         CONTINUE
***avant:    call iswap(itab(i),itab(j))
             IAUX=ITAB(I)
             ITAB(I)=ITAB(J)
             ITAB(J)=IAUX

             IR=J-1
             GOTO 3
         ENDIF
*---         exploration terminee , on a 2 partitions a trier
         IF(I.EQ.J .AND. ITAB(J).LT.LPIVOT) IR=J
*--          on met le pivot a sa place definitive
***avant:          call iswap(itab(ia),itab(ir))
         IAUX=ITAB(IA)
         ITAB(IA)=ITAB(IR)
         ITAB(IR)=IAUX
*---         on empile la partition la +grande,si elle contient 2 elts
         IRA=IR-1
         IRB=IR+1
         IF(IR-IA .GT. IB-IR) THEN
            IF(IA.LT.IRA) THEN
***avant:                call pilall(ia,ira)
               IF(KOPF >= LGPILE ) THEN
                  STOP 'IORDO :PILES SATURéES 2'
               ENDIF
               KOPF=KOPF+1
               IPIL1(KOPF)=IA
               IPIL2(KOPF)=IRA
            ENDIF
            IA=IRB
          ELSE
            IF(IRB.LT.IB) THEN
***avant:                call pilall(irb,ib)
               IF(KOPF >= LGPILE ) THEN
                  STOP 'IORDO :PILES SATURéES 2'
               ENDIF
               KOPF=KOPF+1
               IPIL1(KOPF)=IRB
               IPIL2(KOPF)=IB
            ENDIF
            IB=IRA
         ENDIF
         GOTO 2
      ENDIF

      IF(KOPF > 0 ) GOTO 1

*
*-----   le tri est decroissant
      IF(IOP.LT.0) THEN
         NN=N/2
         J=N
           DO I=1,NN
           IAUX=ITAB(I)
           ITAB(I)=ITAB(J)
           ITAB(J)=IAUX
           J=J-1
         END DO
      ENDIF
*
*-----   on elimine les doubles (si abs(iop)=1 )
      IF(IABS(IOP).EQ.1) THEN
         NN=1
         DO I=2,N
           IF(ITAB(I).EQ.ITAB(NN)) CYCLE
           NN=NN+1
           IF(NN.NE.I) ITAB(NN)=ITAB(I)
         END DO
*-----       n est modifie
         N=NN
      ENDIF
*
      END SUBROUTINE IORDO_HOARE

*=====================================================================

      SUBROUTINE IORDO_BUCKET(N,ITAB,IOP)
*
*     ----------------------------------------------------------------
*
*     tri rapide utilisant le tri casier ordre croissant ou decroissant
*
*                                                      p.  galon 03-2006
*     ----------------------------------------------------------------
*
*             n        : longueur de itab
*             itab     : tableau a modifier
*          croissant
*             iop      : =0  les valeurs identiques sont consecutives
*                        =1  on vire les doubles ( n est modifie )
*          decroissant
*             iop      : =-2 les valeurs identiques sont consecutives
*                        =-1 on vire les doubles ( n est modifie )
*             si n .le. 1     : retour sans modification
*
* remarque : cet algorithme est performant pour n grand, disons > 100000
* et lorsque le nombre d'entiers entre le min et le max de itab est
* inferieur à nln n. la complexiteé de l'algorithme est toujours de l'or
* de o(n+m) ou m est le cardinal de ntrav (max -min + 1)
*
      IMPLICIT NONE
*
*
      INTEGER, INTENT(INOUT) :: N,ITAB(*)
      INTEGER, INTENT(IN)    :: IOP
*
      INTEGER :: I, NN,IAUX,  LONG, MIN, MAX, J, ICONT
      INTEGER , ALLOCATABLE :: NTRAV(:)
*
*
      IF(N < 2) RETURN
*
* --- valeurs mini et maxi du tableau et longueur
*
      MIN = MINVAL(ITAB(1:N))
      MAX = MAXVAL(ITAB(1:N))
      LONG = MAX - MIN + 1
*
      ALLOCATE (NTRAV(LONG))
*
* --- initialisation du tableau
*
      NTRAV(:) = 0
*
* --- on compte les occurences de chaque entier
*
      DO I = 1, N
         J = ITAB(I) - MIN + 1
         NTRAV(J) = NTRAV(J) + 1
      ENDDO
*
* --- on reconstruit le tableau initial
*     (dans l'ordre croissant)
*
      ICONT = 0
      DO I = 1, LONG
*
        DO WHILE (NTRAV(I) > 0)
          ICONT = ICONT + 1
          ITAB(ICONT) = MIN + I - 1
          NTRAV(I) = NTRAV(I) - 1
        END DO
*
      END DO
      IF( N /= ICONT) THEN
         STOP 'IORDO_BUCKET ERROR 1'
      ENDIF
*
      DEALLOCATE (NTRAV)
*
*-----   le tri est decroissant
*
      IF(IOP < 0) THEN
         NN=N/2
         J=N
           DO I=1,NN
           IAUX=ITAB(I)
           ITAB(I)=ITAB(J)
           ITAB(J)=IAUX
           J=J-1
         END DO
      ENDIF
*
*
*-----   on elimine les doubles (si abs(iop)=1 )
*
      IF(IABS(IOP).EQ.1) THEN
         NN=1
         DO I=2,N
           IF(ITAB(I) == ITAB(NN)) CYCLE
           NN=NN+1
           IF(NN.NE.I) ITAB(NN)=ITAB(I)
         END DO
!-----       N est modifie
         N=NN
      ENDIF
*
      END SUBROUTINE IORDO_BUCKET
```

epx_pass_1.pl

```
#-------------------------------------------------------------------------------
# First pass LaTeX manual generation
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$base = "gbnotice.ttx";
$out  = "gbnotice.tex";
$log  = "epx_pass_1.log";
$del  = "rm -f";
$errfil = "epx_pass_1.err";
#-------------------------------------------------------------------------------
# Verify that the template (unfiltered input) file is present
```

```
#
if (! -r $base) {
  $errmsg = "File $base not found!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Remove log file if already present
#
if (-r $log) {
  system ("$del $log");
}
#-------------------------------------------------------------------------------
# Remove auxiliary files
#
system ("epx_clean_manual_files");
#-------------------------------------------------------------------------------
# Filter the base file (by using the local standard keys plus the LATEX1 key)
# This generates the LaTeX file for the first pass
#
system ("epx_filter_manual -q -k LATEX1 $base");
#-------------------------------------------------------------------------------
# Run LaTeX
#
if (system ("latex --halt-on-error $out >>$log 2>&1")) {
  $errmsg = "die epx_pass_1: ERROR(s) in latex compilation!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Generate the index file
#
$outb = $out;
$outb =~ s/.tex$//;
if (system ("makeindex $outb 2>>$log")) {
  $errmsg = "die epx_pass_1: ERROR(s) in makeindex!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
exit;


#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_pass_1_pdf.pl

```
#-------------------------------------------------------------------------------
# First pass LaTeX manual generation (PDF)
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$base = "gbnotice.ttx";
$out  = "gbnotice.tex";
$log  = "epx_pass_1_pdf.log";
$del  = "rm -f";
$errfil = "epx_pass_1_pdf.err";
#-------------------------------------------------------------------------------
# Verify that the template (unfiltered input) file is present
#
if (! -r $base) {
  $errmsg = "File $base not found!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Remove log file if already present
#
if (-r $log) {
  system ("$del $log");
}
#-------------------------------------------------------------------------------
# Remove auxiliary files
#
system ("epx_clean_manual_files");
#-------------------------------------------------------------------------------
# Filter the base file (by using the local standard keys, plus the LATEX1
#                       and the PDFLATEX keys)
# This generates the LaTeX file for the first PDF pass
#
system ("epx_filter_manual -q -k LATEX1 -k PDFLATEX $base");
#-------------------------------------------------------------------------------
# Run PDFLaTeX
#
if (system ("pdflatex --halt-on-error $out >>$log 2>&1")) {
  $errmsg = "die epx_pass_1_pdf: ERROR(s) in pdflatex compilation!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Generate the index file
#
$outb = $out;
$outb =~ s/.tex$//;
if (system ("makeindex $outb 2>>$log")) {
  $errmsg = "die epx_pass_1_pdf: ERROR(s) in makeindex!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
exit;


#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
```

```
#-------------------------------------------------------------------------------
```

epx_pass_2.pl

```
#-------------------------------------------------------------------------------
# Second pass LaTeX manual generation
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$base = "gbnotice.ttx";
$out  = "gbnotice.tex";
$log  = "epx_pass_2.log";
$del  = "rm -f";
$errfil = "epx_pass_2.err";
#-------------------------------------------------------------------------------
# Verify that input file is present
#
if (! -r $base) {
  $errmsg = "File $base not found!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Remove log file if already present
#
if (-r $log) {
  system ("$del $log");
}
#-------------------------------------------------------------------------------
# Filter the base file (by using the local standard keys only)
# This generates the LaTeX file for the second pass
#
system ("epx_filter_manual -q $base");
#-------------------------------------------------------------------------------
# Run LaTeX
#
if (system ("latex --halt-on-error $out >>$log 2>&1")) {
  $errmsg = "die epx_pass_2: ERROR(s) in latex compilation!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Generate the index file
#
$outb = $out;
$outb =~ s/.tex$//;
if (system ("makeindex $outb 2>>$log")) {
  $errmsg = "die epx_pass_2: ERROR(s) in makeindex!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
exit;


#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_pass_2_pdf.pl

```
#-------------------------------------------------------------------------------
# Second pass LaTeX manual generation (PDF)
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$base = "gbnotice.ttx";
$out  = "gbnotice.tex";
$log  = "epx_pass_2_pdf.log";
$del  = "rm -f";
$errfil = "epx_pass_2_pdf.err";
#-------------------------------------------------------------------------------
# Verify that input file is present
#
if (! -r $base) {
  $errmsg = "File $base not found!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Remove log file if already present
#
if (-r $log) {
  system ("$del $log");
}
#-------------------------------------------------------------------------------
# Filter the base file (by using the local standard keys plus the PDFLATEX key)
# This generates the LaTeX file for the second PDF pass
#
system ("epx_filter_manual -q -k PDFLATEX $base");
#-------------------------------------------------------------------------------
# Run PDFLaTeX
#
if (system ("pdflatex --halt-on-error $out >>$log 2>&1")) {
  $errmsg = "die epx_pass_2_pdf: ERROR(s) in pdflatex compilation!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Generate the index file
#
$outb = $out;
$outb =~ s/.tex$//;
```

```
if (system ("makeindex $outb 2>>$log")) {
  $errmsg = "die epx_pass_2_pdf: ERROR(s) in makeindex!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
exit;


#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_pass_3.pl

```
#-------------------------------------------------------------------------------
# Third pass LaTeX manual generation
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$base = "gbnotice.ttx";
$out  = "gbnotice.tex";
$log  = "epx_pass_3.log";
$tthlog = "tth.log";
$del  = "rm -f";
$ren  = "mv -f";
$errfil = "epx_pass_3.err";
#-------------------------------------------------------------------------------
# Verify that input file is present
#
if (! -r $base) {
  $errmsg = "File $base not found!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Remove log file if already present
#
if (-r $log) {
  system ("$del $log");
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  $errmsg = "The EUROPLEXUS environment variable is undefined!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  $errmsg = "The EUROPLEXUS directory: $epx does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Set and check existence of manuals directory
#
$manfildir = "$epx\\manual_filtered";
#
if ( ! -d $manfildir) {
  $errmsg = "The EUROPLEXUS directory: $manfildir does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Filter the base file (by using the local standard keys only)
# This generates the LaTeX file for the third pass
#
system ("epx_filter_manual -q $base");
#-------------------------------------------------------------------------------
# Run LaTeX
#
if (system ("latex --halt-on-error $out >>$log")) {
  $errmsg = "die epx_pass_3: ERROR(s) in latex compilation!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Run DviPS to obtain PostScript version
#
$outb = $out;
$outb =~ s/.tex$//;
if (system ("dvips $outb >>$log 2>>&1")) {
  $errmsg = "die epx_pass_3: ERROR(s) in dvips!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Run Tth to obtain HTML version
#
# FC: NOTE I am obliged to create (and then remove) a subdirectory tth
# and to copy to it all input files because tth does not seem to recognize
# the switch (-p) that tells it to search input files from another dir ...
#
system ("mkdir tth");
system ("cp -p $manfildir\\*.tex tth");
system ("cp -p *.tex tth");
system ("cp -p $outb.* tth");
system ("echo The log file for Tth is on $tthlog >>$log");
chdir "tth";
if (system ("tth <$outb.tex >$outb.html -L$outb 2>>$tthlog")) {
  $errmsg = "die epx_pass_3: ERROR(s) in tth!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
chdir "..";
system ("cp -p tth\\$outb.html .");
system ("cp -p tth\\$tthlog .");
system ("rm -f tth\\*");
system ("rmdir tth");
#-------------------------------------------------------------------------------
# Rename the files
#
system ("$ren $outb.dvi manual.dvi >>$log");
print "      The DVI manual file is               : manual.dvi.\n";
system ("$ren $outb.ps manual.ps >>$log");
print "      The PostScript manual file is        : manual.ps.\n";
system ("$ren $outb.html manual.html >>$log");
print "      The HTML (Tth) manual file is        : manual.html.\n";
#-------------------------------------------------------------------------------
exit;


#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_pass_3_pdf.pl

```
#-------------------------------------------------------------------------------
# Third pass LaTeX manual generation (PDF)
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$base = "gbnotice.ttx";
$out  = "gbnotice.tex";
$log  = "epx_pass_3_pdf.log";
$del  = "rm -f";
$ren  = "mv -f";
$errfil = "epx_pass_3_pdf.err";
#-------------------------------------------------------------------------------
# Verify that input file is present
#
if (! -r $base) {
  $errmsg = "File $base not found!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Remove log file if already present
#
if (-r $log) {
  system ("$del $log");
}
#-------------------------------------------------------------------------------
# Filter the base file (by using the local standard keys plus the PDFLATEX key)
# This generates the LaTeX file for the third PDF pass
#
system ("epx_filter_manual -q -k PDFLATEX $base");
#-------------------------------------------------------------------------------
# Run PDFLaTeX
#
if (system ("pdflatex --halt-on-error $out >>$log 2>&1")) {
  $errmsg = "die epx_pass_3_pdf: ERROR(s) in pdflatex compilation!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Rename PDF file to manual file
#
$outb = $out;
$outb =~ s/.tex$//;
system ("$ren $outb.pdf manual.pdf >>$log");
print "      The PDF manual file is               : manual.pdf.\n";
#-------------------------------------------------------------------------------
exit;


#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_pass_4.pl

```
#-------------------------------------------------------------------------------
# Fourth pass LaTeX manual generation (HEVEA + HACHA)
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$base = "gbnotice.ttx";
$out  = "gbnotice.tex";
$log  = "epx_pass_4.log";
$del  = "rm -f";
$ren  = "mv -f";
$mv   = "mv -f";
$errfil = "epx_pass_4.err";
#-------------------------------------------------------------------------------
# Verify that input file is present
#
if (! -r $base) {
  $errmsg = "File $base not found!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Remove .aux file if already present!!! (else hacha fails!!!)
```

```
#
$outb = $out;
$outb =~ s/.tex$//;
#
if (-r $outb.aux) {
  system ("$del $outb.aux");
}
#-------------------------------------------------------------------------------
# Remove log file if already present
#
if (-r $log) {
  system ("$del $log");
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  $errmsg = "The EUROPLEXUS environment variable is undefined!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  $errmsg = "The EUROPLEXUS directory: $epx does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Set and check existence of manuals directory
#
$manfildir = "$epx\\manual_filtered";
#
if ( ! -d $manfildir) {
  $errmsg = "The EUROPLEXUS directory: $manfildir does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Filter the base file (by using the local standard keys plus thye HEVEA key)
# This generates the LaTeX file for the fourth pass
#
system ("epx_filter_manual -q -k HEVEA $base");
#-------------------------------------------------------------------------------
# Run HEVEA
#
if (system ("hevea -v -I $manfildir $out >>$log 2>>&1")) {
  $errmsg = "die epx_pass_4: ERROR(s) in hevea!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Rename single-piece HTML file to manual file
#
system ("$ren $outb.html manual_h.html >>$log");
print "      The hevea monolithic manual file is  : manual_h.html.\n";
#-------------------------------------------------------------------------------
# Run HACHA
#
if (system ("hacha manual_h.html >>$log 2>>&1")) {
  $errmsg = "die epx_pass_4: ERROR(s) in hacha!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Move split HTML file to new local hacha directory
#
system ("rmdir /s/q hacha >>$log 2>>&1");
system ("mkdir hacha >>$log 2>>&1");
system ("$mv index.html hacha >>$log 2>>&1");
system ("$mv manual_h*.html hacha >>$log 2>>&1");
system ("$mv hacha\\manual_h.html . >>$log 2>>&1");
system ("$mv previous_motif.gif hacha >>$log 2>>&1");
system ("$mv next_motif.gif hacha >>$log 2>>&1");
system ("$mv contents_motif.gif hacha >>$log 2>>&1");
print "      The hevea split manual file starts at: hacha\\index.html.\n";
#-------------------------------------------------------------------------------
exit;


#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_read_pin.f

```
      PROGRAM epx_read_pin
      IMPLICIT NONE
      CHARACTER*51 :: header
      INTEGER :: nsteps, step, n_raw, n_ncol, n_rcel, n_rebo
      REAL(8) :: t
      INTEGER :: n_raw_tot, n_raw_max, n_step_contact
      REAL(8) :: n_raw_ave
      nsteps = 0
      n_raw_tot = 0 ! cumulative number of raw contacts
      n_raw_max = 0 ! max number of raw contacts in a step
      n_step_contact = 0 ! number of steps with at leas 1 raw contact
      READ(5,1000,END=100) header
 1000 FORMAT(A)
    1 READ(5,1001,END=100) step, t, n_raw, n_ncol, n_rcel, n_rebo
 1001 FORMAT (I7, E12.5, 1X, 4I7)
      nsteps = nsteps + 1
      IF (n_raw > 0) THEN
        n_raw_tot = n_raw_tot + n_raw
        n_raw_max = MAX (n_raw_max, n_raw)
        n_step_contact = n_step_contact + 1
      ENDIF
      GO TO 1
  100 IF (n_step_contact > 0) THEN
        n_raw_ave = DBLE (n_raw_tot) / DBLE (n_step_contact)
      ELSE
        n_raw_ave = 0.0
      ENDIF
      WRITE (6,1002) nsteps, n_raw_tot, n_raw_max, n_step_contact,
     >               n_raw_ave
 1002 FORMAT (' number of steps read                          =',I12,/,
     >        ' cumulative number of raw contacts             =',I12,/,
     >        ' max number of raw contacts in a step          =',I12,/,
     >        ' number of steps with at least 1 raw contact   =',I12,/,
     >        ' average n. of raw contacts per contacting step =',
     >        1PE12.5)
      END PROGRAM epx_read_pin
```

epx_save.pl

```
#-------------------------------------------------------------------------------
# Save current version of EUROPLEXUS
#-------------------------------------------------------------------------------
# Redirect STDOUT and STDERR to save.log (Fixed path!!!!)
#
$logfile = "C:\\EUROPLEXUS\\save\\save.log";
open STDOUT, ">>$logfile";
open STDERR, ">>$logfile";
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 0
#
if ($#ARGV >= 0) {                        # Index of last argument must be = -1
  $errmsg = "Usage: epx_save\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  $errmsg = "The EUROPLEXUS environment variable is undefined!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  $errmsg = "The EUROPLEXUS directory: $epx does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check existence of evolution version file
#
$evonumfile = "$epx\\VERSION.txt";         # Evolution version file
if (! -r $evonumfile) {
  die "The evolution number file $evonumfile does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Set and check existence of auxiliary directories
#
$savdir = "$epx\\Save";                    # Directory for saved data
if ( ! -d $savdir) {
  die "The EUROPLEXUS directory: $savdir does not exist!\007\n";
}
print "===> Saving current version of EUROPLEXUS sources\n";
#-------------------------------------------------------------------------------
# Make sure we are in $savdir (so we may use . instead of $savdir)
#
$curdir = `pwd`;
chop $curdir;
if ( $curdir ne $savdir ) {
  print "Changing directory to $savdir.\n";
  chdir "$savdir" ||
  die "Can't chdir to $savdir!\007\n";
}
#-------------------------------------------------------------------------------
# Set and check existence of target directories
#
$srcdir = "$epx\\source";
$sccdir = "$epx\\source_c";
$scbdir = "$epx\\blas_lapack_source";
$scsdir = "$epx\\splib_source";
$jrcdir = "$epx\\jrc";
$incdir = "$epx\\include";
$mandir = "$epx\\manual";
$bendir = "$epx\\bench";
$be2dir = "$epx\\bench2";
$hisdir = "$epx\\history";
$exedir = "$epx\\exe";
$trcdir = "$epx\\trace";
$utidir = "$epx\\util";
$inidir = "$epx\\init";
$iqwdir = "$epx\\init_quickwin";
$valdir = "$epx\\validate";
#
if ( ! -d $srcdir) {
  die "The EUROPLEXUS directory: $srcdir does not exist!\007\n";
}
if ( ! -d $sccdir) {
  die "The EUROPLEXUS directory: $sccdir does not exist!\007\n";
}
if ( ! -d $scbdir) {
  die "The EUROPLEXUS directory: $scbdir does not exist!\007\n";
}
if ( ! -d $scsdir) {
  die "The EUROPLEXUS directory: $scsdir does not exist!\007\n";
}
if ( ! -d $jrcdir) {
  die "The EUROPLEXUS directory: $jrcdir does not exist!\007\n";
}
if ( ! -d $incdir) {
  die "The EUROPLEXUS directory: $incdir does not exist!\007\n";
}
if ( ! -d $mandir) {
```

```
  die "The EUROPLEXUS directory: $mandir does not exist!\007\n";
}
if ( ! -d $bendir) {
  die "The EUROPLEXUS directory: $bendir does not exist!\007\n";
}
if ( ! -d $be2dir) {
  die "The EUROPLEXUS directory: $be2dir does not exist!\007\n";
}
if ( ! -d $hisdir) {
  die "The EUROPLEXUS directory: $hisdir does not exist!\007\n";
}
if ( ! -d $exedir) {
  die "The EUROPLEXUS directory: $exedir does not exist!\007\n";
}
if ( ! -d $trcdir) {
  die "The EUROPLEXUS directory: $trcdir does not exist!\007\n";
}
if ( ! -d $utidir) {
  die "The EUROPLEXUS directory: $utidir does not exist!\007\n";
}
if ( ! -d $inidir) {
  die "The EUROPLEXUS directory: $inidir does not exist!\007\n";
}
if ( ! -d $iqwdir) {
  die "The EUROPLEXUS directory: $iqwdir does not exist!\007\n";
}
if ( ! -d $valdir) {
  die "The EUROPLEXUS directory: $valdir does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Read the number of the current evolution
#
$evonum = `cat $evonumfile`;
chop $evonum;
$evonum =~ s/ //g;
print "===> The current evolution index is  : \# $evonum\n";
#-------------------------------------------------------------------------------
# Build 4-digit version of evolution number (with leading zeroes
# if needed)
#
if ($evonum <= 9) {
  $evonum4 = "000$evonum";
}
elsif ($evonum <= 99) {
  $evonum4 = "00$evonum";
}
elsif ($evonum <= 999) {
  $evonum4 = "0$evonum";
}
else {
  $evonum4 = $evonum;
}
#-------------------------------------------------------------------------------
# Produce the tar file and gzip it, if not already present (in this case
# do nothing, since it means that the code has not evolved since last
# save)
#
$tarfile = "epx_$evonum4\.tar";
$tgzfile = "$tarfile\.gz";
if (! -r $tgzfile) {
  $cmd = "tar cvf $tarfile";
#
# IMPORTANT!!! In specifying composite file names for tar, always use
# forward slashes (/) as separators in place of backslashes (\), else the
# command gives strange errors (some apparently random file names
# are not interpreted correctly ...)
#
  $cmd = "$cmd ../source/*.ff";
  $cmd = "$cmd ../source/obsolete/*.ff";
  $cmd = "$cmd ../source_c/*.c";
  $cmd = "$cmd ../source_c/*.pl";
  $cmd = "$cmd ../source_c/*.bat";
  $cmd = "$cmd ../blas_lapack_source/libblas/*.f";
  $cmd = "$cmd ../blas_lapack_source/liblapack/*.f";
  $cmd = "$cmd ../blas_lapack_source/*.pl";
  $cmd = "$cmd ../blas_lapack_source/*.bat";
  $cmd = "$cmd ../splib_source/*.f";
  $cmd = "$cmd ../splib_source/*.pl";
  $cmd = "$cmd ../splib_source/*.bat";
  $cmd = "$cmd ../splib_source/*.txt";
  $cmd = "$cmd ../jrc/source/*.ff";
  $cmd = "$cmd ../jrc/source/*.pl";
  $cmd = "$cmd ../jrc/source/*.bat";
  $cmd = "$cmd ../include/*.inc";
  $cmd = "$cmd ../include/obsolete/*.inc";
  $cmd = "$cmd ../manual/*.ttx";
  $cmd = "$cmd ../manual/obsolete/*.ttx";
  $cmd = "$cmd ../jrc/manual/*.ttx";
  $cmd = "$cmd ../jrc/manual/*.pl";
  $cmd = "$cmd ../jrc/manual/*.bat";
  $cmd = "$cmd ../bench/*.epx";
  $cmd = "$cmd ../bench/*.msh";
  $cmd = "$cmd ../bench/*.zip";
  $cmd = "$cmd ../bench/obsolete/*.epx";
  $cmd = "$cmd ../bench/obsolete/*.msh";
  $cmd = "$cmd ../bench/obsolete/*.zip";
  $cmd = "$cmd ../bench2/*.epx";
  $cmd = "$cmd ../bench2/*.msh";
  $cmd = "$cmd ../bench2/*.zip";
  $cmd = "$cmd ../bench2/*.adt";
  $cmd = "$cmd ../bench2/*.adv";
  $cmd = "$cmd ../jrc/bench/*.epx";
  $cmd = "$cmd ../jrc/bench/*.msh";
  $cmd = "$cmd ../jrc/bench/*.zip";
  $cmd = "$cmd ../history/*.his";
  $cmd = "$cmd ../history/obsolete/*.his";
  $cmd = "$cmd ../exe/europlexus.exe";
  $cmd = "$cmd ../trace/*.log";
  $cmd = "$cmd ../util/*.pl";
  $cmd = "$cmd ../util/*.bat";
  $cmd = "$cmd ../util/*.f";
  $cmd = "$cmd ../util/*.exe";
  $cmd = "$cmd ../init/*.*";
  $cmd = "$cmd ../init_quickwin/*.*";
  $cmd = "$cmd ../validate/*.vld";
  $cmd = "$cmd ../validate/*.zip";
  $cmd = "$cmd ../validate/obsolete/*.vld";
  $cmd = "$cmd ../validate/obsolete/*.zip";
# print "$cmd\n";
  system ("$cmd");
  system ("gzip -9 $tarfile");
}
else {
  print "File $tgzfile is already in $savdir: the code has not evolved!\n";
}
#
exit;


#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_schtasks.pl

```
system("schtasks /create /tn epx_evo /
tr C:\\EUROPLEXUS\\Util\\epx_evol_start.bat /sc daily         /st 00:30:00 /
ru folco /rp fol1412%");
system("schtasks /create /tn epx_sav /
tr C:\\EUROPLEXUS\\Util\\epx_save.bat       /sc weekly /d SUN /st 12:00:00 /
ru folco /rp fol1412%");
system("schtasks /create /tn epx_chk /
tr C:\\EUROPLEXUS\\Util\\epx_evol_check.bat /sc daily         /st 18:30:00 /
ru folco /rp fol1412%");
```

epx_schtasks_64.pl

```
#system("schtasks /create /tn t_exe /
tr E:\\EUROPLEXUS\\Util\\epx_ftp_putexe.bat /sc once /st 12:40:00 /ru folco /
rp fol1412%");
#system("schtasks /create /tn t_man /
tr E:\\EUROPLEXUS\\Util\\epx_ftp_putman.bat /sc once /st 12:41:00 /ru folco /
rp fol1412%");
system("schtasks /create /tn epx_evo /
tr E:\\EUROPLEXUS\\Util\\epx_evol_64.bat     /sc daily         /st 01:00:00 /
ru folco /rp fol1412%");
#system("schtasks /create /tn epx_sav /
tr E:\\EUROPLEXUS\\Util\\epx_save.bat       /sc weekly /d SUN /st 01:00:00 /
ru folco /rp fol1412%");
#system("schtasks /create /tn epx_chk /
tr E:\\EUROPLEXUS\\Util\\epx_evol_check.bat /sc daily         /st 06:00:00 /
ru folco /rp fol1412%");
system("schtasks /create /tn bk_64_D    /
tr C:\\backup_64bit_D.bat                 /sc daily         /st 06:30:00 /
ru folco /rp fol1412%");
system("schtasks /create /tn bk_64_E    /
tr C:\\backup_64bit_E.bat                 /sc daily         /st 07:00:00 /
ru folco /rp fol1412%");
system("schtasks /create /tn bk_epx_srv /
tr C:\\backup_epx_server_64.bat           /sc daily         /st 08:00:00 /
ru folco /rp fol1412%");
```

epx_schtasks_test.pl

```
system("schtasks /create /tn epx_evo_test /
tr C:\\EUROPLEXUS\\Util\\epx_evol_test.bat /sc daily /st 10:08:00 /ru folco /
rp fol1412%");
```

epx_setat.bat

```
at 07:00 /every:M,T,W,Th,F,S,Su cmd /c epx_evol_start
at 11:00 /every:Su cmd /c epx_save
```

epx_setvars.bat

```
@echo off

rem This version of the epx_setvars script should be reasonably platform-indepen
dent.
rem It requires INTEL the Fortran Compiler 11 to be installed (environment varia
ble
rem IFORT_COMPILER11 should be set).
rem It should automatically detect whether the machine is 32 or 64 bit by using
rem the environment variable PROCESSOR_ARCHITECTURE (which should be set
rem either to x86 or to AMD64.

if "%EPX_VARS_SET%" == "OK" goto End

echo Setting the variables ...

rem I am obliged to comment out the following safety tests because they
rem fail with 64-bit compilers (although they would work OK on 32-bit)
rem due to the fact that under 64-bit the directory name in the environment
rem variable IFORT_COMPLIER11 is
rem    C:\Program Files (x86)\ ...
rem The spaces in the directory name do not cause any problem, but the
rem parentheses (especially the closed parenthesis) do!!!
rem If the file name contains a parenthesis, then the exist command
rem does not work. The call command works if I put a double set of "
rem around the file name (see below), but that does not help with
rem the exists command ...
rem To solve the problem with parantheses, one could try to escape them
rem by the ^ character, but it seems too complicated and I did not try it.

rem IF exist "%IFORT_COMPILER11%" (

  SET IFV="%IFORT_COMPILER11%Bin\ifortvars.bat"

rem ) ELSE (
```

```
rem echo ERROR : directory IFORT_COMPILER11 = '%IFORT_COMPILER11%' does not exis
t !
rem exit /B 1
rem )

rem IF not exist "%IFV%" (
rem   echo ERROR : ifortvars.bat script does not exist !
rem   exit /B 1
rem )

IF "%PROCESSOR_ARCHITECTURE%" == "x86" (
  @call ""%IFV%"" ia32
  SET EPX_VARS_SET=OK
  exit /B 0
)

IF "%PROCESSOR_ARCHITECTURE%" == "AMD64" (
  @call ""%IFV%"" intel64
  SET EPX_VARS_SET=OK
  exit /B 0
)

echo ERROR : wrong value of environment variable PROCESSOR_ARCHITECTURE = '%PROC
ESSOR_ARCHITECTURE%' !
echo         accepted values are : x86 or AMD64
exit /B 1

:End
```

epx_test_benchmarks.pl

```
#-------------------------------------------------------------------------------
# Test EUROPLEXUS benchmark test file(s)
#-------------------------------------------------------------------------------
use File::Basename;                         # Gives access to fileparse function
use HTTP::Date;
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$errs = 0;
$copy = "cp -p";
$del = "rm -f";
$errfil = "epx_test_benchmarks.err";
$exe = "-e epx.exe";
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 0 or 1 or 2 or 3
#
if ($#ARGV > 2) {                   # Index of last argument must be <= 2
  $errmsg = "Usage: epx_test_benchmarks [-e <exec>[.exe]] [-l] [-s]\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#       -e     use exectutable <exec>[.exe] NOT local one (epx.exe)
#       -l     perform local tests only, NOT local + standard ones
#       -s     use standard executable, NOT local one (epx.exe)
#
while ($ARGV[0] =~ /^-/) {
  $_ = shift;
  if (/^-l(.*)/) {
    $loc = "yes";
    printf "Command line switch: local test only\n";
  }
  elsif (/^-s(.*)/) {
    $stdexe = "";
    printf "Command line switch: use standard executable\n";
  }
  elsif (/^-e$/) {
    $_ = shift;
    $exe = "-e $_";
    printf "Command line switch: use $exe\n";
  }
  else {
    $errmsg = "ERROR: unknown switch: $_\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  $errmsg = "The EUROPLEXUS environment variable is undefined!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  $errmsg = "The EUROPLEXUS directory: $epx does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS bench directory exists
#
$epxb = "$epx\\bench";
if ( ! -d $epxb) {
  $errmsg = "The EUROPLEXUS bench directory: $epxb does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Delete  old output file
#
if (-f 'epx_test_benchmarks.bad') {
   system('rm epx_test_benchmarks.bad');
}
#-------------------------------------------------------------------------------
# If the standard executable was chosen, do a (single) temporary copy
#
if (defined $stdexe) {
  #
  # Copy the standard executable to the %TEMP% directory under a unique name
  #
  # Check that the temporary directory exists
  $temp = $ENV{'TEMP'};
  if ( ! -d $temp) {
    $errmsg = "The EUROPLEXUS temporary directory: $temp does not exist!\007\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
  $stdexe = "$epx\\exe\\europlexus.exe";
  # Get current date and time
  #
  $time = time;                       # Machine time (seconds since epoch)
  $gmttime = time2str($time);         # Universal (GMT) date and time
  $gmttime =~ s/...../;
  $gmttime =~ s/ GMT//;
  $gmttime =~ s/ /_/;
  $gmttime =~ s/ /_/;
  $gmttime =~ s/ /_/;
  $gmttime =~ s/:/_/;
  $gmttime =~ s/:/_/;
  $tempexe = "$temp\\epx_$gmttime.exe";
  print "$copy $stdexe $tempexe\n";
  system("$copy $stdexe $tempexe");
  $exe = "-e $tempexe";
}
#-------------------------------------------------------------------------------
# Loop 1: execute all local tests, if any
#
@FileNames = glob ("*.epx");
if ($#FileNames >= 0) {
  print "Executing local tests:\n";
  foreach $file (@FileNames) {
    $base = $file;
    $base =~ s/.epx//;
    #
    unlink "epx_bench.err";
    system ("epx_bench $exe -l -b $base");
    if ( -r "epx_bench.err" ) {
      $errs = $errs + 1;
      system ("cat epx_bench.err >> epx_test_benchmarks.bad");
    }
    #
    # Clean up: delete or rename fort.*
    #
    unlink "fort.9";
    unlink "fort.15";
    if (-r "fort.16") {rename ("fort.16", "$base.listing")};
    if (-r "fort.24") {rename ("fort.24", "$base.ps")};
  }
}
else {
  print "No local tests are present.\n";
}
#-------------------------------------------------------------------------------
# Loop 2: execute those standard tests that are not replaced by a local
# tests (same name), but only if the -l switch has not been given!
#
if (defined($loc)) {
  print "Standard tests are not performed, as requested.\n";
}
else {
  print "Executing standard tests, from directory $epxb:\n";
#
# Problem in perl 6.0 : the globbing with composite file names
# containing \ characters and wildcards (*) does not seem to work
# (it worked with previous versions). To avoid the problem,
# replace all \ with /.
#
  $epxbfiles = $epxb;               # set it equal to dir name (with \)
  $epxbfiles =~ s/\\/\//g;          # replace each \ by /
  $epxbfiles = "$epxbfiles/*.epx";  # add wildcard
  @FileNames = glob ("$epxbfiles"); # now the globbing should work ...
# print "(Files: $epxbfiles)\n";
  foreach $file (@FileNames) {
    ($base,$path,$suffix) = fileparse($file,"\.epx");
    $loc = "$base\.epx";
    if (! -r $loc) {
      #
      unlink "epx_bench.err";
      system ("epx_bench $exe -b $base");
      if ( -r "epx_bench.err" ) {
        $errs = $errs + 1;
        system ("cat epx_bench.err >> epx_test_benchmarks.bad");
      }
    }
  }
}
#-------------------------------------------------------------------------------
if ($errs > 0) {
  $errmsg = "Died epx_test_benchmarks: $errs run failure(s)!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
else {
  print "Benchmarks suite has succesfully terminated.\n";
}
#
# Delete the temporary executable if any
#
if (defined $tempexe) {
  unlink $tempexe;
}
#
exit;


#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_test_evo.pl

```
#-------------------------------------------------------------------------------
# Test EUROPLEXUS benchmark test file(s)
#-------------------------------------------------------------------------------
use File::Basename;                        # Gives access to fileparse function
use HTTP::Date;
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$errs = 0;
$copy = "cp -p";
$del = "rm -f";
#-------------------------------------------------------------------------------
# Any number of arguments is accepted (including 0)
#
if ($#ARGV < -1) {                         # Index of last argument must be >= -1
  $errmsg = "Usage: epx_test_evo [-c]\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#       -c       check all at run-time
#
if ($#ARGV >= 0) { # There is at least one argument
  while ($ARGV[0] =~ /^-/) {
    $_ = shift;
    if (/^-c(.*)/) {
      $check = "yes";
      printf "Command line switch: check\n";
    }
    else {
      $errmsg = "ERROR: unknown switch: $_\n";
      &ERRFIL ($errfil, $errmsg); die "$errmsg";
    }
  }
}

#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  $errmsg = "The EUROPLEXUS environment variable is undefined!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  $errmsg = "The EUROPLEXUS directory: $epx does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Delete  old output file
#
if (-f 'epx_evo_test.err') {
   system('rm epx_evo_test.err');
}
#remove test if existing
if ( -d "test" ) {
    system("rm -f test/*.*");
}
else {
  system("mkdir test");
}
#
# evo.tgz unzip
#
print "Unzip evo.tgz\n";
system("cp evo.tgz test");
chdir "test" || die "Can't chdir to test!\007\n";
system("gunzip evo.tgz");
system("tar xvf evo.tar");
#
# epx_newer
#
print "Test epx_newer\n";
system("epx_newer > _epx_newer.txt");
#test, if epx_newer contains Newer
open (OFILE, "_epx_newer.txt");
while (<OFILE>) {
  chop;
  $fil = $_;
  print "$fil\n";
  if(index($fil,"Newer") >= 0) {
    print "Newer file detected!STOP!\n";
    exit;
  }
}
#
# take users
#
print "Take dependencies\n";
@FileNames = glob ("m_*.ff");
if ($#FileNames >= 0) {
  foreach $file (@FileNames) {
    system("epx_modtree $file");
    system("epx_get_users");
  }
}
#
# Compiling, linking, benchmarks
#
print "Compiling, linking, benchmarks\n";
if($check){
  system("epx_cmp -c");
  system("epx_lk -c");
}
else {
  system("epx_cmp -o");
  system("epx_lk -o");
}
system("epx_test_benchmarks");

exit;


#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_test_includes.pl

```
#-------------------------------------------------------------------------------
# Test EUROPLEXUS include file(s)
#-------------------------------------------------------------------------------
use File::Basename;                        # Gives access to fileparse function
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$rout = "_includers.txt";
$temp = "_temp_.txt";
$copy = "cp -p";
$del  = "rm -f";
$errs = 0;
$errfil = "epx_test_includes.err";
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 0
#
if ($#ARGV != -1) {                        # Index of last argument must be = -1
  $errmsg = "Usage: epx_test_includes\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
$gnudir = $ENV{'GNUDIR'};
if ( ! defined $gnudir) {
  $errmsg = "The GNUDIR environment variable is undefined!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
$sort = "$gnudir\\sort.exe"; # Full path needed! Else uses MS-DOS's sort!
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  $errmsg = "The EUROPLEXUS environment variable is undefined!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  $errmsg = "The EUROPLEXUS directory: $epx does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Build array of include file names on current directory
#
@FileNames = glob ("*.inc");
#-------------------------------------------------------------------------------
# Build list of includers (source files that include any of the include files)
#
system ("$del $rout");
#
print "Building the includers list ...\n";
foreach $file (@FileNames) {
  print "  Searching includers of $file\n";
  system ("epx_grep -l \"$file\" >>$rout");
}
#-------------------------------------------------------------------------------
# Sort list of includers by eliminating multiple name occurrences
#
system ("$sort -u $rout >$temp");
system ("$del $rout");
#system ("ren $temp $rout");
system ("$copy $temp $rout");
system ("$del $temp");
$num = `cat $rout | wc -l`;
chop $num;
$num =~ s/ //g;
print "... includers list built and sorted ($num files)\n";
#-------------------------------------------------------------------------------
# Loop 1a on routine(s) to be retrieved and compiled:
# treat only MODULE files (m_*.ff) (Compilation errors are tolerated)
#
$pref = "$epx\\source";
print "Compiling modules (1), if any (Errors allowed):\n";
open (ROUTS, $rout);
while (<ROUTS>) {
  chop;
  $file = $_;
  #-----------------------------------------------------------------------------
  # Set up file name
  #
  ($name,$path,$suffix) = fileparse($file,"\.ff");
  $_ = $name;
  if ( m/^m_/ ) {
    #---------------------------------------------------------------------------
    # File name is a module name (m_*.ff)
    #
    $filnam = "$name$suffix";
    #---------------------------------------------------------------------------
    if ( -r $filnam ) {
      #-------------------------------------------------------------------------
      # Module file is already on current dir. Just compile it
      #
```

```
      system ("epx_cmp -q -o $name");
    }
    else {
      #-------------------------------------------------------------------------
      # Module file is not on current dir. Retrieve it, compile it and
      # then delete it (only the .obj and .mod are updated)
      #
      print "Retrieving ... ";
      system ("epx_get -q $name");
      system ("epx_cmp -q -o $name");
      system ("$del $filnam");
    }
  }
}
close ROUTS;
#-------------------------------------------------------------------------------
# Loop 1b on routine(s) to be retrieved and compiled:
# treat only MODULE files (m_*.ff) (Compilation errors are tolerated)
#
$pref = "$epx\\source";
print "Compiling modules (2), if any (Errors allowed):\n";
open (ROUTS, $rout);
while (<ROUTS>) {
  chop;
  $file = $_;
  #-----------------------------------------------------------------------------
  # Set up file name
  #
  ($name,$path,$suffix) = fileparse($file,"\.ff");
  $_ = $name;
  if ( m/^m_/ ) {
    #---------------------------------------------------------------------------
    # File name is a module name (m_*.ff)
    #
    $filnam = "$name$suffix";
    #---------------------------------------------------------------------------
    if ( -r $filnam ) {
      #-------------------------------------------------------------------------
      # Module file is already on current dir. Just compile it
      #
      system ("epx_cmp -q -o $name");
    }
    else {
      #-------------------------------------------------------------------------
      # Module file is not on current dir. Retrieve it, compile it and
      # then delete it (only the .obj and .mod are updated)
      #
      print "Retrieving ... ";
      system ("epx_get -q $name");
      system ("epx_cmp -q -o $name");
      system ("$del $filnam");
    }
  }
}
close ROUTS;
#-------------------------------------------------------------------------------
# Loop 2 on routine(s) to be retrieved and compiled:
# treat all files (Compilation errors are NOT tolerated)
#
$pref = "$epx\\source";
print "Compiling all files (errors NOT allowed):\n";
open (ROUTS, $rout);
while (<ROUTS>) {
  chop;
  $file = $_;
  #-----------------------------------------------------------------------------
  # Set up file name
  #
  ($name,$path,$suffix) = fileparse($file,"\.ff");
  $filnam = "$name$suffix";
  #-----------------------------------------------------------------------------
  if ( -r $filnam ) {
    #---------------------------------------------------------------------------
    # Routine file is already on current dir. Just compile it
    #
    unlink "epx_cmp.err";
    system ("epx_cmp -q -o $name");
    if ( -r "epx_cmp.err" ) {$errs = $errs + 1};
  }
  else {
    #---------------------------------------------------------------------------
    # Routine file is not on current dir. Retrieve it, compile it and
    # then delete it (only the .obj is updated)
    #
    print "Retrieving ... ";
    system ("epx_get -q $name");
    #
    unlink "epx_cmp.err";
    system ("epx_cmp -q -o $name");
    if ( -r "epx_cmp.err" ) {$errs = $errs + 1};
    #
    system ("$del $filnam");
  }
}
close ROUTS;
#
if ($errs > 0) {
  $errmsg = "Died epx_test_includes: $errs compilation failure(s)!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
else {
  print "Includes testing has successfully terminated.\n";
}
#
exit;


#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_test_man.pl

```
#-------------------------------------------------------------------------------
# Test  a single local file from the EUROPLEXUS manual
#
$base0 = "driver_base.ttx";
$base = "driver_base.tex";
$out = "driver.tex";
$copy = "cp -p";
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 1
#
if ($#ARGV != 0) {                          # Index of last argument must be = 0
  $errmsg = "Usage: epx_test_man file<.tex>\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# The argument is the (base) name of the TEX file
#
$name0 = $ARGV[0];
#-------------------------------------------------------------------------------
# Set up file name
#
$name0 =~ s/\.ttx$//;                       # Remove '.ttx' extension if present
$name = "$name0.tex";                       # Build name of filtered file
$name0 = "$name0.ttx";                      # Add '.ttx' extension
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  $errmsg = "The EUROPLEXUS environment variable is undefined!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  $errmsg = "The EUROPLEXUS directory: $epx does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Set and check existence of manuals directory
#
$mandir = "$epx\\manual";
#
if ( ! -d $mandir) {
  $errmsg = "The EUROPLEXUS directory: $mandir does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Retrieve LaTeX manual "driver" template file
#
unlink $base0;
unlink $base;
unlink $out;
print "  Retrieving template file $base0 from $mandir.\n";
system ("$copy $mandir\\$base0 .");
print "$copy $mandir\\$base0 .\n";
#-------------------------------------------------------------------------------
# Filter the base file (by using the local standard keys)
#
print "  Filtering template file $base0.\n";
system ("epx_filter_manual $base0");
#-------------------------------------------------------------------------------
# Filter the input file (by using the local standard keys)
#
print "  Filtering input file $name0.\n";
system ("epx_filter_manual $name0");
#-------------------------------------------------------------------------------
# Test the file
#
open (INPUT, $base);
open (OUTPUT, "> $out");
while (<INPUT>) {
  s/^\\input\{file\.tex\}/\\input\{$name\}/;
  print (OUTPUT);
}
close OUTPUT;
close INPUT;
#
system ("latex $out");
#-------------------------------------------------------------------------------
exit;


#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_test_manuals.bat

```
@rem = '--*-Perl-*--
@echo off
if "%OS%" == "Windows_NT" goto WinNT
perl -x -S "%0" %1 %2 %3 %4 %5 %6 %7 %8 %9
goto endofperl
:WinNT
perl -x -S "%0" %*
if NOT "%COMSPEC%" == "%SystemRoot%\system32\cmd.exe" goto endofperl
if %errorlevel% == 9009 echo You do not have Perl in your PATH.
if errorlevel 1 goto script_failed_so_exit_with_non_zero_val 2>nul
goto endofperl
@rem ';
#!perl
#line 15
#-------------------------------------------------------------------------------
# Test EUROPLEXUS manual file(s)
#-------------------------------------------------------------------------------
```

```
use File::Basename;                     # Gives access to fileparse function
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$copy = "cp -p";
$del  = "rm -f";
$base = "gbnotice.ttx";
$errfil = "epx_test_manuals.err";
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 0
#
if ($#ARGV != -1) {                     # Index of last argument must be = -1
  $errmsg = "Usage: epx_test_manuals\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  $errmsg = "The EUROPLEXUS environment variable is undefined!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  $errmsg = "The EUROPLEXUS directory: $epx does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Set and check existence of manuals directory
#
$mandir = "$epx\\manual";
#
if ( ! -d $mandir) {
  $errmsg = "The EUROPLEXUS directory: $mandir does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Build array of LaTeX source file names on current directory
#
@FileNames = glob ("*.ttx");
#-------------------------------------------------------------------------------
# Retrieve LaTeX manual template file if it is not already in current dir
#
if (-r $base) {
  print "  Template file $base is already in current dir.\n";
}
else {
  print "  Retrieving template file $base from $mandir.\n";
  system ("$copy $mandir\\$base .");
}
#-------------------------------------------------------------------------------
# Filter all local LaTeX source files (by using the local standard keys)
#
print "  Filtering all local LaTeX source files:\n";
foreach $file (@FileNames) {
  $_ = $file;
  print "    $file\n";
  system ("epx_filter_manual -q $file");
}
#-------------------------------------------------------------------------------
# Make the manuals
#
print "  Making DVI, PostScript, HTML (Tth) and HTML (Hevea/Hacha) versions:\n";
#
print "    Pass 1.\n";
unlink "epx_pass_1.err";
system("epx_pass_1");
if ( -r  "epx_pass_1.err" ) {
  $errmsg = "epx_test_manuals, ERROR in pass_1!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#
print "    Pass 2.\n";
unlink "epx_pass_2.err";
system("epx_pass_2");
if ( -r  "epx_pass_2.err" ) {
  $errmsg = "epx_test_manuals, ERROR in pass_2!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#
print "    Pass 3.\n";
unlink "epx_pass_3.err";
system("epx_pass_3");
if ( -r  "epx_pass_3.err" ) {
  $errmsg = "epx_test_manuals, ERROR in pass_3!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
print "    Pass 4.\n";
unlink "epx_pass_4.err";
system("epx_pass_4");
if ( -r  "epx_pass_4.err" ) {
  $errmsg = "epx_test_manuals, ERROR in pass_4!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#
print "  Making PDF version:\n";
#
print "    Pass 1 PDF.\n";
unlink "epx_pass_1_pdf.err";
system("epx_pass_1_pdf");
if ( -r  "epx_pass_1_pdf.err" ) {
  $errmsg = "epx_test_manuals, ERROR in pass_1_pdf!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#
print "    Pass 2 PDF.\n";
unlink "epx_pass_2_pdf.err";
system("epx_pass_2_pdf");
if ( -r  "epx_pass_2_pdf.err" ) {
  $errmsg = "epx_test_manuals, ERROR in pass_2_pdf!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#
print "    Pass 3 PDF.\n";
unlink "epx_pass_3_pdf.err";
system("epx_pass_3_pdf");
if ( -r  "epx_pass_3_pdf.err" ) {
  $errmsg = "epx_test_manuals, ERROR in pass_3_pdf!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
print "Manuals testing has successfully terminated.\n";
#
exit;


#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------

__END__
:endofperl
```

epx_test_manuals.pl

```
#-------------------------------------------------------------------------------
# Test EUROPLEXUS manual file(s)
#-------------------------------------------------------------------------------
use File::Basename;                     # Gives access to fileparse function
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$copy = "cp -p";
$del  = "rm -f";
$base = "gbnotice.ttx";
$errfil = "epx_test_manuals.err";
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 0
#
if ($#ARGV != -1) {                     # Index of last argument must be = -1
  $errmsg = "Usage: epx_test_manuals\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  $errmsg = "The EUROPLEXUS environment variable is undefined!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  $errmsg = "The EUROPLEXUS directory: $epx does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Set and check existence of manuals directory
#
$mandir = "$epx\\manual";
#
if ( ! -d $mandir) {
  $errmsg = "The EUROPLEXUS directory: $mandir does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Build array of LaTeX source file names on current directory
#
@FileNames = glob ("*.ttx");
#-------------------------------------------------------------------------------
# Retrieve LaTeX manual template file if it is not already in current dir
#
if (-r $base) {
  print "  Template file $base is already in current dir.\n";
}
else {
  print "  Retrieving template file $base from $mandir.\n";
  system ("$copy $mandir\\$base .");
}
#-------------------------------------------------------------------------------
# Filter all local LaTeX source files (by using the local standard keys)
#
print "  Filtering all local LaTeX source files:\n";
foreach $file (@FileNames) {
  $_ = $file;
  print "    $file\n";
  system ("epx_filter_manual -q $file");
}
#-------------------------------------------------------------------------------
# Make the manuals
#
print "  Making DVI, PostScript, HTML (Tth) and HTML (Hevea/Hacha) versions:\n";
#
print "    Pass 1.\n";
unlink "epx_pass_1.err";
system("epx_pass_1");
if ( -r  "epx_pass_1.err" ) {
  $errmsg = "epx_test_manuals, ERROR in pass_1!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#
print "    Pass 2.\n";
unlink "epx_pass_2.err";
system("epx_pass_2");
```

```
if ( -r  "epx_pass_2.err" ) {
  $errmsg = "epx_test_manuals, ERROR in pass_2!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#
print "    Pass 3.\n";
unlink "epx_pass_3.err";
system("epx_pass_3");
if ( -r  "epx_pass_3.err" ) {
  $errmsg = "epx_test_manuals, ERROR in pass_3!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
print "    Pass 4.\n";
unlink "epx_pass_4.err";
system("epx_pass_4");
if ( -r  "epx_pass_4.err" ) {
  $errmsg = "epx_test_manuals, ERROR in pass_4!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#
print "  Making PDF version:\n";
#
print "    Pass 1 PDF.\n";
unlink "epx_pass_1_pdf.err";
system("epx_pass_1_pdf");
if ( -r  "epx_pass_1_pdf.err" ) {
  $errmsg = "epx_test_manuals, ERROR in pass_1_pdf!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#
print "    Pass 2 PDF.\n";
unlink "epx_pass_2_pdf.err";
system("epx_pass_2_pdf");
if ( -r  "epx_pass_2_pdf.err" ) {
  $errmsg = "epx_test_manuals, ERROR in pass_2_pdf!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#
print "    Pass 3 PDF.\n";
unlink "epx_pass_3_pdf.err";
system("epx_pass_3_pdf");
if ( -r  "epx_pass_3_pdf.err" ) {
  $errmsg = "epx_test_manuals, ERROR in pass_3_pdf!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
print "Manuals testing has successfully terminated.\n";
#
exit;


#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_test_sources.pl

```
#-------------------------------------------------------------------------------
# Test EUROPLEXUS source file(s) :
#  - if epx_ordo.txt is present in current dir, compile sources just once,
#                                               in the given order
#  - else, generate the file by invoking epx_ordo, then proceed as above
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$errs = 0;
$errfil = "epx_test_sources.err";
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 0
#
if ($#ARGV != -1) {                       # Index of last argument must be = -1
  $errmsg = "Usage: epx_test_sources\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  $errmsg = "The EUROPLEXUS environment variable is undefined!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  $errmsg = "The EUROPLEXUS directory: $epx does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
$ordo = "epx_ordo.txt";
if ( ! -r $ordo ) {
  print "The file $ordo is missing: generate it!\n";
  system ("epx_ordo");
}
else {
  print "The file $ordo is present!\n";
}
#-------------------------------------------------------------------------------
# Build array of source file names from file epx_ordo.txt
#
open (ORDO, $ordo);
while (<ORDO>) {
  chop;
  $fil = $_;
  if ( ! -r $fil ) {
    $errmsg = "Died epx_test_sources: $fil missing!\007\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
  push @FileNames, $_;
}
close ORDO;
#-------------------------------------------------------------------------------
# Loop 2: compile all source files (Errors NOT tolerated)
#
print "Compiling all files in the right order (errors NOT allowed):\n";
foreach $file (@FileNames) {
  $base = $file;
  $base =~ s/\.ff//;
  #
  unlink "epx_cmp.err";
  system ("epx_cmp -q -o $base");
  if ( -r "epx_cmp.err" ) {$errs = $errs + 1};
}
#
if ($errs > 0) {
  $errmsg = "Died epx_test_sources: $errs compilation failures(s)!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
else {
  print "Sources testing has successfully terminated.\n";
}
#
exit;


#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_update_header.pl

```
# Update "header" of EUROPLEXUS source file(s) by replacing it
# with the header of the current standard version of the file(s)
#-------------------------------------------------------------------------------
# Fixed parameters
#
$temp = "_temp.ff";
#-------------------------------------------------------------------------------
# Default values
#
$ext = "ff";
$dir = "source";
#-------------------------------------------------------------------------------
# Check number of arguments, must be 1 or 2
#
if ($#ARGV < 0 || $#ARGV > 1) {          # Index of last argument must be 0 or 1
  die "Usage: epx_update_header [-i] [-b] [-m] [-v] name(s)<.extension>\007\n";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#       -i     include file, i.e. ".inc" file, not source (.ff) file
#       -b     benchmark, i.e. ".epx" file, not source (.ff) file
#       -m     manual, i.e. ".ttx" file, not source (.ff) file
#       -v     validation, i.e. ".vld" file, not source (.ff) file
#
while ($ARGV[0] =~ /^-/) {
  $_ = shift;
  if (/^-i(.*)/) {
    $ext = "inc";
    $dir = "include";
    printf "Command line switch: include file\n";
  }
  elsif (/^-b(.*)/) {
    $ext = "epx";
    $dir = "bench";
    printf "Command line switch: bench file\n";
  }
  elsif (/^-m(.*)/) {
    $ext = "ttx";
    $dir = "manual";
    printf "Command line switch: manual file\n";
  }
  elsif (/^-v(.*)/) {
    $ext = "vld";
    $dir = "validate";
    printf "Command line switch: validation file\n";
  }
  else {
    die "ERROR: unknown switch: $_\n";
  }
}
#-------------------------------------------------------------------------------
# Verify that temporary file is not already present
#
if (-r $temp) {
  print "File $temp already present in current directory\n";
  print "Please, rename it or delete it\n";
  exit;
}
#-------------------------------------------------------------------------------
# Check if the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  die "The EUROPLEXUS environment variable is undefined!\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  die "The EUROPLEXUS directory: $epx does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# All arguments are file names: do the globbing
#
@FileNames = glob (join ($",@ARGV));
```

```
#-------------------------------------------------------------------------------
# Loop on file(s) to be processed
#
foreach $name (@FileNames) {
  #-----------------------------------------------------------------------------
  # Set up file name
  #
  $base = $name;
  $base =~ s/\.$ext$//;                         # Remove extension if present
  $file = "$base.$ext";
  print "Filtering: $file\n";
  #-----------------------------------------------------------------------------
  # Verify that the relevant file exists in the EUROPLEXUS directory
  #
  $from = "$epx\\$dir\\$file";               # Full file name
  if ( ! -r $from ) {
    die "File $file does not exist in $epx\\$dir!\007\n";
  }
  #
  # Process File
  #
  $fromhead = `head -1 $from`;
  #$filehead = `head -1 $file`;
  open (INPUT, "type $file |");
  open (TEMP, ">> $temp");
  $line = 0;
  while (<INPUT>) {
    $line = $line + 1;
    if ( $line == 1 ) {
      $_ = $fromhead;
    }
    print (TEMP);
  }
  close TEMP;
  close INPUT;
  system("del $file");
  system("ren $temp $file");
}
exit;
```

epx_vali.pl

```
#-------------------------------------------------------------------------------
# Verify results of EUROPLEXUS benchmark test:
#   If the stderr file contains "ARRET NORMAL" or "NORMAL END" then:
#     If the stderr file contains "VALIDATION: ERREUR" or
#                                 "VALIDATION: ERROR"  then WRONG;
#                                                      (qualification error)
#     Else if the stderr file contains "VALIDATION: CORRECT" or
#                                      "VALIDATION: OK" then OK;
#                                                      (qualification OK)
#     Else OK; (normal end without qualification)
#   Else WRONG. (abnormal end)
#
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$errfil = "epx_vali.err";
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 1
#
if ($#ARGV != 0) {                        # Index of last argument must be = 0
  $errmsg = "Usage: epx_vali basename[.epx]\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# The argument is the (base) name of the test
#
$name = $ARGV[0];
#-------------------------------------------------------------------------------
# Set up file name
#
$base = $name;
$base =~ s/\.epx$//;                      # Remove '.epx' extension if present
$std = "$base.std";                       # stderr file
#-------------------------------------------------------------------------------
# Verify that stderr file exists
#
if ( ! -r $std ) {
  $errmsg = "The stderr file: $std does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
print "Verifying benchmark: $base --> ";
#-------------------------------------------------------------------------------
open (STD, $std);
while (<STD>) {
  if ( m/ARRET NORMAL/ || m/NORMAL END/  ) {
    $foundan = 1;
  }
}
close STD;
#-------------------------------------------------------------------------------
if ( ! defined ($foundan)) {
  $val = 0;
  $reason = "Abnormal end";
}
else {
  open (STD, $std);
  while (<STD>) {
    if ( m/VALIDATION: ERREUR/ || m/VALIDATION: ERROR/ ) {
      $val = 0;
      $foundqe = 1;
      $reason = "Qualification error";
    }
  }
  close STD;
  if ( ! defined ($foundqe)) {
    $val = 1;
    $reason = "Normal end without qualification";
    open (STD, $std);
    while (<STD>) {
      if ( m/VALIDATION: CORRECT/ || m/VALIDATION: OK/ ) {
        $val = 1;
        $reason = "Qualification OK";
      }
    }
    close STD;
  }
}
#-------------------------------------------------------------------------------
print "$reason\.\n";
#
if ($val != 1) {
  $note = "* * * * * * * * * * * * * * * * * * * * *";
  print "\n";
  $errmsg = "Wrong validation for $base ($reason) $note !\007\n\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
exit;


#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_validate.pl

```
#-------------------------------------------------------------------------------
# Execute EUROPLEXUS validation test
#-------------------------------------------------------------------------------
# Modules needed
#
use HTTP::Date;
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$copy = "cp -p";
$del = "rm -f";
$errfil = "epx_validate.err";
#-------------------------------------------------------------------------------
# Check number of arguments, must be 1 (there are no optional switches)
#
if ($#ARGV < 0 || $#ARGV > 0) {     # Index of last argument must be 0
  $errmsg = "Usage: epx_validate basename[.vld]\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  $errmsg = "The EUROPLEXUS environment variable is undefined!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS validate sub-directory exists
#
$epxb = "$epx\\validate";
if ( ! -d $epxb) {
  $errmsg = "The EUROPLEXUS validate directory: $epxb does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# The argument is the (base) name of the test
#
$name = $ARGV[0];
#-------------------------------------------------------------------------------
# Set up file name
#
$base = $name;
$base =~ s/\.vld$//;                          # Remove '.vld' extension if present
#-------------------------------------------------------------------------------
# Verify that the .vld and .zip files exists in the appropriate dir
#
$vld = "$base.vld";
$zip = "$base.zip";
$epxvld = "$epxb\\$vld";
$epxzip = "$epxb\\$zip";
if ( ! -r $epxvld ) {
  $errmsg = "File $vld does not exist in directory $epxb!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
if ( ! -r $epxzip ) {
  $errmsg = "File $zip does not exist in directory $epxb!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Verify that local subdirectory $base does NOT exist, then create it
#
$curdir = `pwd`;
chop $curdir;
if ( -e $base ) {
  $errmsg = "Subdirectory $base already exists in $curdir!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
system ("mkdir $base");
#-------------------------------------------------------------------------------
# Temporarily move to $base subdirectory
#
print "\nChanging directory to $base.\n";
chdir "$base" || die "Can't chdir to $base!\007\n";
#-------------------------------------------------------------------------------
# Copy .vld and .zip files from the EPX validate dir to current dir,
# if necessary
#
system("$copy $epxvld .");
```

```
system("$copy $epxzip .");
#-------------------------------------------------------------------------------
# Verify that the main input (and mesh) file do not exist in the current dir
#
$inp = "$base.epx";
$msh = "$base.msh";
if ( -r $inp ) {
  $errmsg = "File $inp exists already in current directory!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
if ( -r $msh ) {
  $errmsg = "File $msh exists already in current directory!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Extract contents from the .zip file (the .zip file remains untouched)
# (-o overwrites any pre-existing files)
system("unzip -o $zip");
#-------------------------------------------------------------------------------
# Verify that the (main) input file exists in the current dir
#
$inp = "$base.epx";
$msh = "$base.msh";
if ( ! -r $inp ) {
  $errmsg = "File $inp does not exist in zip file!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Remove any result files which might come from the .ZIP file
#
system("del *.listing");
system("del *.avi");
system("del *.bmp");
system("del *.ps");
#-------------------------------------------------------------------------------
# Build array of EUROPLEXUS input file names on current directory
#
@FileNames = glob ("*.epx");
#-------------------------------------------------------------------------------
# Process each input file
foreach $inp (@FileNames) {
  $_ = $inp;
  print "    RUNNING $inp\n";
  #-----------------------------------------------------------------------------
  # Comment out any OPNF PATH directives from the .epx file
  $temp = "_epx_validate.epx";
  open (INPUT, "type $inp |");
  open (TEMP, ">> $temp");
  while (<INPUT>) {
    if ( m/^[ \t]*OPNF[ \t]+PATH[ \t]+/i ) { # i means case-insensitive match!
      s/^/!/;
    }
    print (TEMP);
  }
  close TEMP;
  close INPUT;
  system("del $inp");
  system("ren $temp $inp");
  #-----------------------------------------------------------------------------
  # Execute the run (in batch mode, so STDOUT is redirected)
  #
  system("epx_bench -l -b $inp");
}
#-------------------------------------------------------------------------------
# Move back to original directory
#
print "\nChanging directory to $curdir.\n";
chdir "$curdir" || die "Can't chdir to $curdir!\007\n";
exit;


#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_validate_all.pl

```
#-------------------------------------------------------------------------------
# Execute a series of EUROPLEXUS validation tests
#-------------------------------------------------------------------------------
# Modules needed
#
use HTTP::Date;
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
# Default values
#
$copy = "cp -p";
$del = "rm -f";
$errfil = "epx_validate_all.err";
$owner = "*";
#-------------------------------------------------------------------------------
# Check number of arguments, must be 0 or 2
#
if ($#ARGV < -1 || $#ARGV > 1) {     # Index of last argument must be -1 or 1
  $errmsg = "Usage: epx_validate_all [-o <owner>]\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#       -o <owner>   specify "owner" of validation tests (all by default)
#
while ($ARGV[0] =~ /^-/) {
  $_ = shift;
  if (/^-o$/) {
    $_ = shift;
    $owner = $_;
    print "Owner: $owner\n";
  }
  else {
    $errmsg = "ERROR: unknown switch: $_\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  $errmsg = "The EUROPLEXUS environment variable is undefined!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS validate sub-directory exists
#
$epxv = "$epx\\validate";
if ( ! -d $epxv) {
  $errmsg = "The EUROPLEXUS validate directory: $epxv does not exist!\007\n";
  &ERRFIL ($errfil, $errmsg); die "$errmsg";
}
#-------------------------------------------------------------------------------
# Build array of EUROPLEXUS validation file names
#
# NOTE : the glob operator seems to work only on the current directory!
#        therefore, I change dir to the validation dir to build up
#        the list of all available .vld files, then come back to current dir
#
$curdir = `pwd`;
chop $curdir;
print "\nChanging directory to $epxv.\n";
chdir "$epxv" || die "Can't chdir to $epxv!\007\n";
print "Owner = $owner\n";
@FileNames = glob ("vl_$owner\_*.vld");
print "\nChanging directory to $curdir.\n";
chdir "$curdir" || die "Can't chdir to $curdir!\007\n";
#-------------------------------------------------------------------------------
# Process each validation file
foreach $vld (@FileNames) {
  $_ = $vld;
  $base = $vld;
  $base =~ s/\.vld$//;
  print "PROCESSING $base\n";
  system("epx_validate $base");
#-------------------------------------------------------------------------------
}
#
exit;


#-------------------------------------------------------------------------------
sub ERRFIL {
  local($errfile, $errmsg) = @_;
  open (EF, ">>$errfile");
  print EF "$errmsg";
  close EF;
}
#-------------------------------------------------------------------------------
```

epx_verif.pl

```
#
# Verify results of EUROPLEXUS benchmark test
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 1
#
if ($#ARGV != 0) {                        # Index of last argument must be = 0
  die "Usage: epx_verif basename[.epx]\007\n";
}
#-------------------------------------------------------------------------------
# The argument is the (base) name of the test
#
$name = $ARGV[0];
#-------------------------------------------------------------------------------
# Set up file name
#
$base = $name;
$base =~ s/\.epx$//;                          # Remove '.epx' extension if present
print "Verifying benchmark: $base\n";
#-------------------------------------------------------------------------------
# Compare the listings
#
system("del coco.txt");
system("compar $base.listing fort.16 >coco.txt");
#-------------------------------------------------------------------------------
# Compare the PostScript files, if any
#
$ps = "$base.ps";
system("del psps.txt");
if ( -r $ps ) {
  system("compar $base.ps fort.24 >psps.txt");
}
#
exit;
```

epx_vi.pl

```
#-------------------------------------------------------------------------------
# Edit (gvim) Fortran ff|inc|his file from EUROPLEXUS library
#-------------------------------------------------------------------------------
# Default values
#
$ext = "ff";                              # By default, source file
$dir = "source";                          # By default, source directory
#-------------------------------------------------------------------------------
# Check number of arguments, must be 1 or 2 or 3
#
if ($#ARGV < 0 || $#ARGV > 2) {       # Index of last argument must be 0, 1 or 2
  die "Usage: epx_vi [-i] [-h] [-b] [-m] [-u] [-l] [-p] name[.<extension>]\007\n
";
}
```

```
#-------------------------------------------------------------------------------
# Process optional switches:
#       -i     include file, i.e. ".inc" file, not source (.ff) file
#       -h     history file, i.e. ".his" file, not source file
#       -b     benchmark, i.e. ".epx" file, not source (.ff) file
#       -m     manual, i.e. ".ttx" file, not source (.ff) file
#       -u     utility, i.e. ".pl" file, not source (.ff) file
#       -l     listing, i.e. ".listing" file, not source (.ff) file
#       -p     postscript, i.e. ".ps" file, not source (.ff) file
#       -v     validation file, i.e. ".vld" file, not source (.ff) file
#
while ($ARGV[0] =~ /^-/) {
  $_ = shift;
  if (/^-i(.*)/) {
    $ext = "inc";
    $dir = "include";
    printf "Command line switch: include file\n";
  }
  elsif (/^-h(.*)/) {
    $ext = "his";
    $dir = "history";
    printf "Command line switch: history file\n";
  }
  elsif (/^-b(.*)/) {
    $ext = "epx";
    $dir = "bench";
    printf "Command line switch: bench file\n";
  }
  elsif (/^-m(.*)/) {
    $ext = "ttx";
    $dir = "manual";
    printf "Command line switch: manual file\n";
  }
  elsif (/^-u(.*)/) {
    $ext = "pl";
    $dir = "util";
    printf "Command line switch: utility (Perl) file\n";
  }
  elsif (/^-l(.*)/) {
    $ext = "listing";
    $dir = "bench";
    printf "Command line switch: listing file\n";
  }
  elsif (/^-p(.*)/) {
    $ext = "ps";
    $dir = "bench";
    printf "Command line switch: PostScript file\n";
  }
  elsif (/^-v(.*)/) {
    $ext = "vld";
    $dir = "validate";
    printf "Command line switch: validation file\n";
  }
  else {
    die "ERROR: unknown switch: $_\n";
  }
}
#-------------------------------------------------------------------------------
# Check if the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  die "The EUROPLEXUS environment variable is undefined!\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  die "The EUROPLEXUS directory: $epx does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Set up file name
#
$name = $ARGV[0];                           # Get fist and only argument
$base = $name;
$base =~ s/\.$ext$//;                       # Remove extension if present
$file = "$base.$ext";
#-------------------------------------------------------------------------------
# Verify that the source|include file exists in the source|include directory
#
$from = "$epx\\$dir\\$file";                # Full file name
if ( ! -r $from ) {
  die "File $file does not exist in $epx\\$dir!\007\n";
}
#-------------------------------------------------------------------------------
# Edit file with gvim (IN READ_ONLY MODE!)
#
#fc NO! the following produces an extra "console" window ...
#       call "vi.bat" instead
#fc system("start gvim -R $from");
system("vi -R $from");
#fc
#
exit;
```

evototgz.pl

```
#
# Read and split "evo.txt" file containing evolution comments.
# Check presence of related files and produce .TGZ package ready for
# evolution.
#
$tempdir = "_tmp";
$evo = "evo.txt";
$lines = 0;
#
# Check number of arguments, must be 0
#
if ($#ARGV != -1) {
  print "Usage: evototgz\007\n";
  exit;
}
#
# Verify that temporary subdirectory is not already present
#
if (-d $tempdir) {
  print "Subdirectory $tempdir already present in current directory\n";
  print "Please, rename it or delete it\n";
  exit;
}
#
# Verify that "evo.txt" file is present
#
if (! -r $evo) {
  print "File $evo not found in current directory\n";
  exit;
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  $errmsg = "The EUROPLEXUS environment variable is undefined!\007\n";
  die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  $errmsg = "The EUROPLEXUS directory: $epx does not exist!\007\n";
  die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS bench directory exists
#
$epxb = "$epx\\bench";
if ( ! -d $epxb ) {
  $errmsg = "The EUROPLEXUS bench directory: $epxb does not exist!\007\n";
  die "$errmsg";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS validations directory exists
#
$epxv = "$epx\\validate";
if ( ! -d $epxv ) {
  $errmsg = "The EUROPLEXUS validations directory: $epxv does not exist!\007\n";
  die "$errmsg";
}
#-------------------------------------------------------------------------------
# Create work subdirectory and copy evo.txt to it
# CD to work subdirectory
#
system ("mkdir $tempdir");
system ("cp -f $evo $tempdir");
chdir "$tempdir" || die "Can't chdir to $tempdir!\007\n";
#-------------------------------------------------------------------------------
# Process comments file
#
open (INPUT, $evo);
#open (TEMP, ">> $temp");
while (<INPUT>) {
  if ( m/^[\ \t]*[a-zA-Z]+[a-zA-Z0-9_]*\.[fF][fF][\ \t]*$/ ) {
    #
    # Fortran source file
    #
    chop;
    $fprev = $fbase;
    $fbase = $_;
    $fbase =~ s/ //g;
    $fbase =~ s/\t//g;
    $fbase =~ tr /[A-Z]/[a-z]/;
    $fbase =~ s/\.ff//;
    $file = "..\\$fbase.ff";
    print "$fbase.ff\n";
    if (-r $file) {
      system ("cp -f $file $fbase.ff");
    }
    else {
      print "Can't find file $fbase.ff\007\n";
      exit;
    }
    if (defined ($opened)) {
      if ($lines > 0) {
        close FILE;
        undef $opened;
      }
      else {
        print "ERROR - no comments for file $fprev.\n";
        exit;
      }
    }
    open (FILE, ">> $fbase.txt");
    $opened = 1;
  }
  elsif ( m/^[\ \t]*[a-zA-Z]+[a-zA-Z0-9_]*\.[tT][tT][xX][\ \t]*$/ ) {
    #
    # Manual source file
    #
    chop;
    $fprev = $fbase;
    $fbase = $_;
    $fbase =~ s/ //g;
    $fbase =~ s/\t//g;
    $fbase =~ tr /[A-Z]/[a-z]/;
    $fbase =~ s/\.ttx//;
    $file = "..\\$fbase.ttx";
    print "$fbase.ttx\n";
    if (-r $file) {
      system ("cp -f $file $fbase.ttx");
    }
    else {
      print "Can't find file $fbase.ttx\007\n";
      exit;
    }
    if (defined ($opened)) {
      if ($lines > 0) {
        close FILE;
        undef $opened;
      }
      else {
        print "ERROR - no comments for file $fprev.\n";
        exit;
      }
    }
```

```
  open (FILE, ">> $fbase.txt");
  $opened = 1;
}
elsif ( m/^[\ \t]*[a-zA-Z]+[a-zA-Z0-9_]*\.[eE][pP][xX][\ \t]*$/ ) {
  #
  # Benchmark source file
  #
  chop;
  $fprev = $fbase;
  $fbase = $_;
  $fbase =~ s/ //g;
  $fbase =~ s/\t//g;
  $fbase =~ tr /[A-Z]/[a-z]/;
  $fbase =~ s/\.epx//;
  $file = "..\\$fbase.epx";
  $fmsh = "..\\$fbase.msh";
  $bmsh = "$epxb\\$fbase.msh";
  $fzip = "..\\$fbase.zip";
  print "$fbase.epx\n";
  if (-r $file) {
    system ("cp -f $file $fbase.epx");
  }
  else {
    print "Can't find file $fbase.epx\007\n";
    exit;
  }
  # if corresponding .MSH exists in the bench directory, then it is
  # mandatory to include it in the evolution !
  if (-r $bmsh) {
    if (-r $fmsh) {
      system ("cp -f $fmsh $fbase.msh");
      print "$fbase.msh\n";
    }
    else {
      $errmsg = "Obligatory mesh file $fbase.msh not found!\007\n";
      die "$errmsg";
    }
  }
  # treat also case of adding a non pre-existing .MSH
  else {
    if (-r $fmsh) {
      system ("cp -f $fmsh $fbase.msh");
      print "$fbase.msh\n";
    }
  }
  # the corresponding .ZIP file, instead, is optional
  if (-r $fzip) {
    system ("cp -f $fzip $fbase.zip");
    print "$fbase.zip\n";
  }
  if (defined ($opened)) {
    if ($lines > 0) {
      close FILE;
      undef $opened;
    }
    else {
      print "ERROR - no comments for file $fprev.\n";
      exit;
    }
  }
  open (FILE, ">> $fbase.txt");
  $opened = 1;
  # print (TEMP);
  # print;
}
elsif ( m/^[\ \t]*[a-zA-Z]+[a-zA-Z0-9_]*\.[vV][lL][dD][\ \t]*$/ ) {
  #
  # Validation source file
  #
  chop;
  $fprev = $fbase;
  $fbase = $_;
  $fbase =~ s/ //g;
  $fbase =~ s/\t//g;
  $fbase =~ tr /[A-Z]/[a-z]/;
  $fbase =~ s/\.vld//;
  $file = "..\\$fbase.vld";
  $bval = "$epxv\\$fbase.zip";
  $fzip = "..\\$fbase.zip";
  print "$fbase.vld\n";
  if (-r $file) {
    system ("cp -f $file $fbase.vld");
  }
  else {
    print "Can't find file $fbase.vld\007\n";
    exit;
  }
  # the corresponding .ZIP file is mandatory
  if (-r $fzip) {
    system ("cp -f $fzip $fbase.zip");
    print "$fbase.zip\n";
  }
  else {
    print "ERROR - Missing .ZIP file $fzip (mandatory for validation).\007\n";
    exit;
  }
  if (defined ($opened)) {
    if ($lines > 0) {
      close FILE;
      undef $opened;
    }
    else {
      print "ERROR - no comments for file $fprev.\n";
      exit;
    }
  }
  open (FILE, ">> $fbase.txt");
  $opened = 1;
}
elsif ( m/^[\ \t]*\n/ ) {
  #
  # Blank line
  #
  if (defined ($opened)) {
    if ($lines > 0) {
      close FILE;
      undef $opened;
      $lines = 0;
    }
    else {
      print "ERROR - no comments for file $fbase.\n";
      exit;
    }
  }
}
else {
  #
  # Comment line
  #
  if (defined ($opened)) {
    print (FILE $_);
    $lines = $lines + 1;
  }
  else {
    print "ERROR - extra comments found (no associated file).\n";
    exit;
  }
}
}
#
# EOF reaached in INPUT
#
if (defined ($opened)) {
  if ($lines > 0) {
    close FILE;
    undef $opened;
    $lines = 0;
  }
  else {
    print "ERROR - no comments for file $fbase.\n";
    exit;
  }
}
close FILE;
close INPUT;
system("rm -f  $evo");
#
print "\nTARring and ZIPping ...\n";
system("tar cvf evo.tar *");
system("gzip -9 evo.tar");
#
print "\nCleaning up ...\n";
system("mv -f evo.tar.gz ..\\evo.tgz");
chdir ".." || die "Can't chdir to parent directory!\007\n";
system("rm -rf $tempdir");
#
print "\n*** The result is in evo.tgz ***\n";
exit;
```

ftp_putexe.pl

```
#-------------------------------------------------------------------------------
# Update executable module on the machine outside the firewall
# INTERACTIVE VERSION (to be launched manually)
#-------------------------------------------------------------------------------
# Modules needed
#
use NET::FTP;
#-------------------------------------------------------------------------------
# Default values
#
$exefile = "europlexus.exe";
$exefilegz = "europlexus.exe.gz";
$copy = "cp -p -f";
$del  = "rm -f";
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 0 or 1
#
if ($#ARGV >= 1) {                    # Index of last argument must be = -1 or 0
  die "Usage: ftp_putexe [-w]\007\n";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#       -w      copy QuickWin executable instead of Console
#
while ($ARGV[0] =~ /^-/) {
  $_ = shift;
  if (/^-w(.*)/) {
    $exefile = "europlexusqw.exe";
    $exefilegz = "europlexusqw.exe.gz";
    printf "Command line switch: QuickWin\n";
  }
  else {
    $errmsg = "ERROR: unknown switch: $_\n";
    &ERRFIL ($errfil, $errmsg); die "$errmsg";
  }
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  die "The EUROPLEXUS environment variable is undefined!\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  die "The EUROPLEXUS directory: $epx does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Set and check existence of auxiliary directories
#
$evodir = "$epx\\Fromcentral";             # Directory used for evolution
if ( ! -d $evodir) {
  die "The EUROPLEXUS directory: $evodir does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Make sure we are in the directory that will send FTP files
#
$curdir = `pwd`;
chop $curdir;
if ( $curdir ne $evodir ) {
  print "\nChanging directory to $evodir.\n";
```

```
  chdir "$evodir" ||
  die "Can't chdir to $evodir!\007\n";
}
#-------------------------------------------------------------------------------
# Set and check existence of file to be copied
#
if ( -r $exefile) {
  die "The EUROPLEXUS executable: $exefile exists already!\007\n";
}
if ( -r $exefilegz) {
  die "The EUROPLEXUS executable: $exefilegz exists already!\007\n";
}
$exe = "$epx\\exe\\$exefile";
system("$copy $exe .");
system("gzip -9 $exefile");
#-------------------------------------------------------------------------------
# Access machine outside the firewall, and copy executable
#
$ftp = Net::FTP->new("europlexus.jrc.it", Debug => 1);
$ftp->login("europlexus","la,mi!");
$ftp->cwd("/srv/sites/plexus/web_site/consortium/exe");
$ftp->binary;
print "Updating file $exefilegz ...\n";
$ftp->delete("$exefilegz");
$ftp->put("$exefilegz");
$ftp->quit;
#-------------------------------------------------------------------------------
# Clean up and exit
#
system("$del $exefilegz");
#
exit;
```

ftp_putman.pl

```
#-------------------------------------------------------------------------------
# Update manuals (html and pdf) on the machine outside the firewall
# INTERACTIVE VERSION (to be launched manually)
#-------------------------------------------------------------------------------
# Modules needed
#
use NET::FTP;
#-------------------------------------------------------------------------------
# Check number of arguments, must be = 0
#
if ($#ARGV != -1) {                       # Index of last argument must be = -1
  die "Usage: ftp_putman\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  die "The EUROPLEXUS environment variable is undefined!\007\n";
}
#-------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  die "The EUROPLEXUS directory: $epx does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Set and check existence of auxiliary directories
#
$mandir = "$epx\\manual";             # Directory containing the manuals
if ( ! -d $mandir) {
  die "The EUROPLEXUS directory: $mandir does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Make sure we are in the directory that will sending FTP files
#
$curdir = `pwd`;
chop $curdir;
if ( $curdir ne $mandir ) {
  print "\nChanging directory to $mandir.\n";
  chdir "$mandir" ||
  die "Can't chdir to $mandir!\007\n";
}
#-------------------------------------------------------------------------------
# Set and check existence of files to be copied
#
$pdffile = "manual.pdf";
if ( ! -r $pdffile) {
  die "The EUROPLEXUS manual (pdf): $pdffile does not exist!\007\n";
}
$htmldir = "hacha";
if ( ! -d $htmldir) {
  die "The EUROPLEXUS manual (html) dir: $htmldir does not exist!\007\n";
}
#-------------------------------------------------------------------------------
# Build up list of files in $htmldir
#
opendir (HACHA, "$htmldir");
@files = readdir (HACHA);
closedir (HACHA);
#-------------------------------------------------------------------------------
# Access machine outside the firewall, and copy manuals
#
$ftp = Net::FTP->new("europlexus.jrc.it", Debug => 1);
$ftp->login("europlexus","la,mi!");
#
# pdf version
#
$ftp->cwd("/srv/sites/plexus/web_site/consortium/manual_pdf");
$ftp->binary;
print "Updating file $pdffile ...\n";
$ftp->delete("$pdffile");
$ftp->put("$pdffile");
#
# html version (hacha)
#
$ftp->cwd("/srv/sites/plexus/web_site/public/manual_html");
print "Updating all files in $htmldir:\n";
#
foreach $file (@files) {
  $_ = $file;
  if ( m/^\./ ) {
    # Do not copy . and .. files !
  }
  else {
    print "  File $file ...\n";
    $fromfile = "$htmldir\\$file";
    $ftp->delete("$file");
    $ftp->put("$fromfile", "$file");
  }
}
#
exit;
```

hello.pl

```
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
$epx = $ENV{'EUROPLEXUS'};
chdir "$epx\\Fromcentral" || die "Can't chdir!\n";
#open (OUT, ">>folco.out");
#print OUT "Hello, world!\n";
#print OUT "EUROPLEXUS is: $epx\n";
#close OUT;
#system("cp -p $epx\\source\\codump.ff \.");
#system("epx_grep -i ipb >ipb.txt");
#system("epx_get celem");
#system("epx_lk -o");
#system("epx_bench bm_cir_bifur");
system("epx_bench -e epx.exe bm_cir_anneau");
#system("hello1");
#die "This is the stderr!\n";
exit;
```

hello1.pl

```
#-------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#-------------------------------------------------------------------------------
$epx = $ENV{'EUROPLEXUS'};
#chdir "$epx\\Fromcentral" || die "Can't chdir!\n";
open (OUT, ">>folco1.out");
print OUT "Hello1, world!\n";
print OUT "EUROPLEXUS1 variable is: $epx\n";
close OUT;
exit;
```

k2000_big.pl

```
#
# Perform the "UNIX expansion" of @ARGV
#
foreach $i (0 .. $#ARGV) {
  if ($i == 0) {
    @ARGU=glob ($ARGV[$i]);
  } else {
    @ARGU=(@ARGU,glob ($ARGV[$i]));
  }
}
@ARGV=(@ARGU);

#
# default parameters
#
#---TO MODIFY EVENTUALLY. TO MODIFY EVENTUALLY---
#$CastemPath= "D:\\Appl\\Visual Castem 2000";
#$CastemPath= "D:\\Appl\\VisualCastem2000";
#fc $CastemPath= "D:\\Appl\\VisualCast3m";
$CastemPath= "C:\\Cast3m";
#fc
$ENV{ESOPE_TEMP} = "D:\\TEMP";
#---TO MODIFY EVENTUALLY. TO MODIFY EVENTUALLY---
$ENV{ESOPE_PARAM} = "ESOPE=312000000";
$ENV{GIBI_FILES} = "$CastemPath\\gibi_files";
$ENV{MIF_PATH} = "$CastemPath\\gibi_files";
$CASTEM_QWIN = "$CastemPath\\bin\\prov_b.exe";
$CASTEM_CONS = "$CastemPath\\bin\\prov_b.exe";
$exten_f= "$esope_temp\\gemat.temp";
#print $CastemPath; print "\n";
#print $CASTEM_QWIN; print "\n";
#print $CASTEM_CONS; print "\n";
#
# input of 3 optionals parameters: size, loadmodule and run directory
#
#       -s size
#       -l load
#       -d dir
#

$sizz = 2;
$size = $sizz * 100000;
#printf "default=$sizz, size=$size\n";
$eso_p = "ESOPE=" . "$size";

$ltrk = 8192;
#$mram = 24000000;
$mram = 312000000;
$mtrk = 300000;

$rdir = ".\\";

while ($ARGV[0] =~ /^-/) {
  $_ = shift;
  if (/^-s(.*)/) {
      $sizz = ($1 ? $1 : shift);
```

```
      $size = $sizz * 100000;
#      printf "command line sizz=$sizz, size=$size\n";
      $rize = $size - $mram;
      if ( $rize > 0 ) {
        $ntrk = int(($rize / $ltrk) + 1);
        if ( $ntrk > $mtrk ) {
          die "to large memory request: $ntrk > $mtrk\n";
        }
        $eso_p = "ESOPE=" . "$mram,LTRK=" . "$ltrk,NTRK=" . "$ntrk";
      }
      else {
        $ntrk = 0;
        $eso_p = "ESOPE=" . "$size";
      }
  }
  elsif (/^-l(.*)/) {
      $load = ($1 ? $1 : shift);
      $CASTEM_QWIN = "$load";
      -e $CASTEM_QWIN or die("$CASTEM_QWIN does not exist!!\n");
  }
  elsif (/^-d(.*)/) {
      $rdir = ($1 ? $1 : shift);
      -e $rdir or die("$rdir does not exist!!\n");
  }
  else {
      die "unrecognized switch: $_\n";
  }
}

#
# we now look for the + parameter
#
$f_null=f_null;

if ($ARGV[0] =~ /^\+/) {

#
#### we are in the interractive part
#

  $file = "";
  $_ = shift;
  if (/^\+(.*)/) {
    $file =  ($1 ? $1 : shift);
  }

#
####### verify the extension of the input name
#

  if ( $file ne "" ) {
    ($ffile = $file) =~ s/\.dgibi//;
    $GibiFile = "$ffile.dgibi";
    -e $GibiFile or die("$GibiFile does not exist !\n");
    $f90 = "fort.90";
    system("copy $GibiFile $f90 1>$f_null");
  }
  else {
    $ffile = "$rdir" . "run";
  }

#
####### do something for the size (if file exists and contains *%siz nnn)
#

  $ENV{ESOPE_PARAM} = "$eso_p";
  if ( $file ne "" ) {
    open (FILE, $file);
    $i=0;
    while (<FILE>) {
      $i = $i + 1;
      if(/\*\%siz\ (.*)/) {
        $sizy = ($1 ? $1 : shift);
#        printf "file sizy=$sizy, sizz=$sizz\n";
        if ($sizy > $sizz){
          $size = $sizy * 100000;
#          printf "size=$size\n";
          $rize = $size - $mram;
          if ( $rize > 0 ) {
            $ntrk = int(($rize / $ltrk) + 1);
            if ( $ntrk > $mtrk ) {
              die "to large memory request: $ntrk > $mtrk\n";
            }
            $eso_q = "ESOPE=" . "$mram,LTRK=" . "$ltrk,NTRK=" . "$ntrk";
          }
          else {
            $ntrk = 0;
            $eso_q = "ESOPE=" . "$size";
          }
          $ENV{ESOPE_PARAM} = "$eso_q";
        }
        last;
      }
      if($i > 5) {
        last;
      }
    }
  }
#
####### query for a possible conflict to open the extension file
#

  if ( $ntrk > 0 ) {
     if ( -e $exten_f ) {
       die "unable to create a SECOND disk extension file $exten_f\n";
     }
  }
#
####### We call K2000 QWIN
#

  system("$CASTEM_QWIN");

#
####### cleaning
#

  -e $f90 and unlink $f90;
  $PLUTO="PLUTO";
  -e $PLUTO and unlink $PLUTO;
  $f98 = "fort.98";
  if ( -e $f98 ) {
     $hisfile = "$ffile.his";
     system("copy $f98 $hisfile 1>$f_null");
     unlink $f98;
  }
#FC  $f24 = "fort.24";
  $f24 = "File.eps";
#FC
  if ( -e $f24 ) {
     $psfile = "$ffile.ps";
     system("copy $f24 $psfile 1>$f_null");
     unlink $f24;
  }
#FC  $f97 = "fort.97";
  $f97 = "File.mif";
#FC
  if ( -e $f97 ) {
     $miffile = "$ffile.mif";
     system("copy $f97 $miffile 1>$f_null");
     unlink $f97;
  }
  if ( $ntrk > 0 ) {
     unlink $exten_f;
  }

}
else {

#
#### we are in batch part
#
####### loop on the input files
#

  foreach $i (0 .. $#ARGV) {
    $file = $ARGV[$i];
    printf "processing $file\n";

#
####### verify the extension
#

    ($ffile = $file) =~ s/\.dgibi//;
    $GibiFile = "$ffile.dgibi";
    if ( -e $GibiFile ) {
      $f90 = "fort.90";
      system("copy $GibiFile $f90 1>$f_null");
      open F90, ">>$f90";
      printf F90 "fin;\n";
      close F90;
#
####### do something for the size (if file exists and contains *%siz nnn)
#

      $ENV{ESOPE_PARAM} = "$eso_p";
      open (FILE, $file);
      $i=0;
      while (<FILE>) {
        $i = $i + 1;
        if(/\*\%siz\ (.*)/) {
          $sizy = ($1 ? $1 : shift);
#          printf "file sizy=$sizy, sizz=$sizz\n";
          if ($sizy > $sizz){
            $size = $sizy * 100000;
#            printf "size=$size\n";
            $rize = $size - $mram;
            if ( $rize > 0 ) {
              $ntrk = int(($rize / $ltrk) + 1);
              if ( $ntrk > $mtrk ) {
                die "to large memory request: $ntrk > $mtrk\n";
              }
              $eso_q = "ESOPE=" . "$mram,LTRK=" . "$ltrk,NTRK=" . "$ntrk";
            }
            else {
              $ntrk = 0;
              $eso_q = "ESOPE=" . "$size";
            }
            $ENV{ESOPE_PARAM} = "$eso_q";
          }
          last;
        }
        if($i > 5) {
          last;
        }
      }
#
####### query for a possible conflict to open the extension file
#

      if ( $ntrk > 0 ) {
         if ( -e $exten_f ) {
           die "unable to create a SECOND disk extension file $exten_f\n";
         }
      }
#
####### We call K2000 CONS (without opening a new window AND in low priority
#                                                              mode)
#

      $prfile = "$ffile.pr";
#fc   system("start /LOW /B /WAIT $CASTEM_CONS <$f90 >$prfile");
#fc start in normal mode, NOT low priority
      system("start      /B /WAIT $CASTEM_CONS <$f90 >$prfile");
#     system("$CASTEM_CONS <$f90 >$prfile");

#
####### cleaning
#

      -e $f90 and unlink $f90;
      $f98 = "fort.98";
      -e $f98 and unlink $f98;
#FC       $f24 = "fort.24";
      $f24 = "File.eps";
#FC
```

```
      if ( -e $f24 ) {
         $psfile = "$ffile.ps";
         system("copy $f24 $psfile 1>$f_null");
         unlink $f24;
      }
#FC      $f97 = "fort.97";
      $f97 = "File.mif";
#FC
      if ( -e $f97 ) {
         $miffile = "$ffile.mif";
         system("copy $f97 $miffile 1>$f_null");
         unlink $f97;
      }
      if ( $ntrk > 0 ) {
         unlink $exten_f;
      }


    }
    else {
      printf "$GibiFile does not exist !\n";
    }
#

  }
}

unlink $f_null;

exit;
```

miktex_folco.ini

```
;; -----------------------------------------------------------------------------
;; LaTeX
;; -----------------------------------------------------------------------------

[LaTeX]

;; Where LaTeX searches for input files.
Input Dirs=.;%R\tex\latex//;%R\tex\generic//;%R\tex//
;E:\EUROPLEXUS\manual_filtered

;; Common LaTeX filename extensions.
Extensions=.tex;.ltx

;; -----------------------------------------------------------------------------
;; pdfLaTeX
;; -----------------------------------------------------------------------------

[pdfLaTeX]

;; Where pdfLaTeX searches for input files.
Input Dirs=.;%R\pdftex\latex//;%R\pdftex\generic//;%R\pdftex//;%R\tex\latex//
;%R\tex\generic//;%R\tex//;E:\EUROPLEXUS\manual_filtered

;; -----------------------------------------------------------------------------
;; pdfTeX
;; -----------------------------------------------------------------------------

[pdfTeX]

;; Where pdfTeX searches for input files.
Input Dirs=.;%R\pdftex\plain//;%R\pdftex\generic//;%R\pdftex//;%R\tex\plain//
;%R\tex\generic//;%R\tex//;E:\EUROPLEXUS\manual_filtered

;; Where pdfTeX searches for PostScript headers.
PSPath=.;%R\pdftex//

;; -----------------------------------------------------------------------------
;; TeX
;; -----------------------------------------------------------------------------

[TeX]

;; Where TeX looks for input files.
Input Dirs=.;%R\tex\plain//;%R\tex\generic//;%R\tex//
;E:\EUROPLEXUS\manual_filtered

;; Common TeX filename extensions.
Extensions=.tex
```

mydir.bat

```
dir >c:\mydir.txt
```

ordo.f

```
      PROGRAM ordo
      IMPLICIT NONE
      INTEGER ::
     >  nt,        ! n. of modules containing at least one USE statement
     >  nlie       ! total n. of USE statements in modules
      CHARACTER(26), ALLOCATABLE ::
     >  list(:),   ! modules containing at least one USE, in alpha order
     >  forwd(:),  ! modules containing USE
     >  bckwd(:)   ! USEd modules
      INTEGER, ALLOCATABLE ::
     >  class(:),  ! levels of modules containing at least one USE statement
     >  liais(:),  ! USE relationships
     >  kbckwd(:),
     >  kforwd(:)
      CHARACTER(26) ::
     >  b,
     >  f
      INTEGER ::
     >  i,
     >  j,
     >  iforwd,
     >  ibckwd,
     >  iliais,
     >  mxi,
     >  iaux,
     >  ier,
     >  lung
      INTEGER, EXTERNAL :: ilyest, norm_class
* read data
      READ *, nt, nlie
      ALLOCATE (list(nt))
      ALLOCATE (forwd(nlie))
      ALLOCATE (bckwd(nlie))
      ALLOCATE (class(nt))
      ALLOCATE (kforwd(nlie))
      ALLOCATE (kbckwd(nlie))
      ALLOCATE (liais(nlie))
      READ *, list
      READ *, (forwd(i), bckwd(i), i = 1, nlie)
* initializations
      class(:)  = 1
      kforwd(:) = 0
      kbckwd(:) = 0
      liais(:)  = 0
* on vire les liaisons dont le bckwd n'est pas dans la liste
      DO i = 1, nlie
        b = bckwd(i)
        iliais = ilyest (nt, list, b)
        liais(i) = iliais
        IF (iliais /= 0) THEN
          kbckwd(i) = iliais
          f = forwd(i)
          kforwd(i) = ilyest (nt, list, f)
        ENDIF
      END DO
*
      mxi = 1
      DO j = 1, nt
        iaux = norm_class (nt, class)
        DO i = 1, nlie
           IF (liais(i) == 0 ) CYCLE
           iforwd = kforwd(i)
           ibckwd = kbckwd(i)
           class(iforwd) = MAX (class(iforwd), class(ibckwd)+1)
           mxi = MAX (mxi, class(iforwd))
        END DO
*
* modifs du 01Aout2005:
* on teste si norm_class ne change pas => convergence
* alors que avant on testait sur la croissance de mxi
*
        ier = norm_class (nt, class)
        IF (ier == iaux) THEN
          ier = 0
          EXIT
        ENDIF
      END DO ! j = 1, nt
*
      IF (ier /= 0) THEN
* ier /= 0 : probleme
        WRITE (0, 1001) ier
        DO j = 1, nt
          IF (class(j) > nt) WRITE (0, 1100) list(j)
        END DO
        CALL report_cycle (nt, nlie, list, forwd, bckwd)
        STOP 'ORDO ERROR : IER /= 0'
      ENDIF
*
      IF (mxi > nt) THEN
* le nbre de classes > nt : probleme
        WRITE (0, 1000) mxi, nt
        DO j = 1, nt
          IF (class(j) > nt) WRITE (0, 1100) list(j)
        END DO
        CALL report_cycle (nt, nlie, list, forwd, bckwd)
        STOP 'ORDO ERROR : MXI > NT'
      ENDIF
*
      DO i = 1, mxi
        DO j = 1, nt
          IF (class(j) == i) THEN
!!            lung = INDEX (list(j), '.ff') + 2
!!            PRINT 2000, list(j)(1:lung)
            PRINT 2000, list(j)
          ENDIF
        END DO
      END DO
*
      STOP 'ORDO : NORMAL END'
*
 1000 FORMAT ('Cyclage: Nbre de classes = ',I4,
     >        ' > nbre de fichiers = ',I4)
 1001 FORMAT ('Cyclage: ier = ',I10)
 1100 FORMAT ('fichiers concernés : ',A)
 2000 FORMAT (A)
*
      END PROGRAM ordo
*=======================================================================
      MODULE m_listused
      IMPLICIT NONE
      PUBLIC :: listused
      TYPE listused
        INTEGER :: nusato
        INTEGER, POINTER :: usato(:)
      END TYPE listused
      TYPE(listused), POINTER :: listu(:)
      END MODULE m_listused
*=======================================================================
      INTEGER FUNCTION ilyest (nt, list, dad)
*
* returns j : > 0  if dad is j-th element of list(:)
*             == 0 if dad is not contained in list(:)
*
      IMPLICIT NONE
* args
      INTEGER, INTENT(IN) :: nt
      CHARACTER(26), INTENT(IN) :: list(nt)
      CHARACTER(26), INTENT(IN) :: dad
*
* locals
      INTEGER :: j
```

```
*
      ilyest = 0
      DO j = 1, nt
        IF (dad == list(j)) THEN
          ilyest = j
          EXIT
        ENDIF
      END DO
*
      END FUNCTION ilyest
*=====================================================================
      INTEGER FUNCTION ilyest_int (nt, list, dad)
*
* returns j : > 0  if dad is j-th element of list(:)
*             == 0 if dad is not contained in list(:)
*
      IMPLICIT NONE
* args
      INTEGER, INTENT(IN) :: nt
      INTEGER, INTENT(IN) :: list(nt)
      INTEGER, INTENT(IN) :: dad
*
* locals
      INTEGER :: j
*
      ilyest_int = 0
      DO j = 1, nt
        IF (dad == list(j)) THEN
          ilyest_int = j
          EXIT
        ENDIF
      END DO
*
      END FUNCTION ilyest_int
*=====================================================================
      INTEGER FUNCTION norm_class (nt, class)
*
* returns the sum of the nt items in class(:)
*
      IMPLICIT NONE
*
      INTEGER, INTENT(IN) :: nt
      INTEGER, INTENT(IN) :: class(nt)
*
* locals
      INTEGER :: ip
*
      norm_class = 0
      DO ip = 1, nt
        norm_class = norm_class + class(ip)
      END DO
*
      END FUNCTION norm_class
*=====================================================================
      SUBROUTINE report_cycle (nmod, nuse, modul, mouse, used)
*
* detect cycle and report it
*
* there are nmod modules each containing at least one USE statement
*
* their names are contained (in alpha order, but this is irrelevant)
* into modul(:)
*
* there are nuse occurrences of the USE statement altogether in these
* nmod modules (therefore it must be nuse >= nmod)
*
* for each USE statement, mouse is the module, used is the used module
* i.e. module mouse contains "USE used".
*
      USE m_listused
*
      IMPLICIT NONE
*
* args
      INTEGER, INTENT(IN) :: nmod, nuse
      CHARACTER(26), INTENT(IN) :: modul(nmod), mouse(nuse), used(nuse)
*
* locals
      INTEGER :: i, m, u, nold, ncycl
      CHARACTER(26) :: mou, usd
      LOGICAL :: lcycle
      INTEGER, POINTER :: cycl(:), done(:)
*
      INTEGER, EXTERNAL :: ilyest
*
      NULLIFY (listu)
      NULLIFY (cycl)
      NULLIFY (done)
*
      ALLOCATE (listu(nmod))
      DO i = 1, nmod
        listu(i)%nusato = 0
        NULLIFY (listu(i)%usato)
      END DO
*
      DO i = 1, nuse
        mou = mouse(i)
        m = ilyest (nmod, modul, mou)
        IF (m == 0) STOP 'REPORT_CYCLE M = 0'
        usd = used(i)
        u = ilyest (nmod, modul, usd)
        IF (u > 0) THEN
          listu(m)%nusato = listu(m)%nusato + 1
        ENDIF
      END DO
*
      DO i = 1, nmod
        IF (listu(i)%nusato > 0) THEN
          ALLOCATE (listu(i)%usato(listu(i)%nusato))
        ENDIF
        listu(i)%nusato = 0
      END DO
*
      DO i = 1, nuse
        mou = mouse(i)
        m = ilyest (nmod, modul, mou)
        IF (m == 0) STOP 'REPORT_CYCLE M = 0'
        usd = used(i)
        u = ilyest (nmod, modul, usd)
        IF (u > 0) THEN
          listu(m)%nusato = listu(m)%nusato + 1
          listu(m)%usato(listu(m)%nusato) = u
        ENDIF
      END DO
*
      DO i = 1, nmod
        IF (listu(i)%nusato > 0) THEN
          nold = listu(i)%nusato
          CALL iordo (listu(i)%nusato, listu(i)%usato, 1) ! vire doubles
          listu(i)%usato(listu(i)%nusato+1:nold) = 0
        ENDIF
      END DO
*
      ALLOCATE (cycl(nmod))
      ALLOCATE (done(nmod))
      done(:) = 0
      DO i = 1, nmod - 1
        IF (done(i) == 1) CYCLE
        IF (listu(i)%nusato == 0) CYCLE
        ncycl = 0
        cycl(:) = 0
        lcycle = .FALSE.
        CALL rec_use (i, nmod, modul, cycl, ncycl, lcycle, done)
        done(i) = 1
        IF (lcycle) EXIT
      END DO
      DEALLOCATE (cycl)
      DEALLOCATE (done)
*
      DEALLOCATE (listu)
*
      END SUBROUTINE report_cycle
*=====================================================================
      RECURSIVE SUBROUTINE rec_use (i, nmod, modul, cycl, ncycl, lcycle,
     >                              done)
*
      USE m_listused
*
      IMPLICIT NONE
*
* args
      INTEGER, INTENT(IN) :: i, nmod
      CHARACTER(26), INTENT(IN) :: modul(nmod)
      INTEGER, INTENT(INOUT) :: ncycl, cycl(nmod), done(nmod)
      LOGICAL, INTENT(OUT) :: lcycle
*
* locals
      INTEGER :: k, p, m, n, rcycl
*
      INTEGER, EXTERNAL :: ilyest_int
*
      IF (listu(i)%nusato == 0) RETURN
      ncycl = ncycl + 1
      cycl(ncycl) = i
*
      DO k = 1, listu(i)%nusato
        n = listu(i)%usato(k)
        p = ilyest_int (ncycl-1, cycl, n)
        IF (p > 0) THEN
          ncycl = ncycl - p + 1
          DO m = 1, ncycl
            cycl(m) = cycl(m+p-1)
          END DO
          cycl(ncycl+1:nmod) = 0
          lcycle = .TRUE.
          CALL print_cycle (ncycl, cycl, nmod, modul)
          RETURN
        ENDIF
        rcycl = ncycl
        CALL rec_use (n, nmod, modul, cycl, rcycl, lcycle, done)
        done(n) = 1
        IF (lcycle) RETURN
      END DO
*
      END SUBROUTINE rec_use
*=====================================================================
      SUBROUTINE print_cycle (ncycl, cycl, nmod, modul)
*
      IMPLICIT NONE
*
* args
      INTEGER, INTENT(IN) :: ncycl, cycl(ncycl), nmod
      CHARACTER(26), INTENT(IN) :: modul(nmod)
*
* locals
      INTEGER :: i, ip1
*
      WRITE(0,1000) ncycl
 1000 FORMAT('Cycle detected in following chain of', I4, ' modules:')
      DO i = 1, ncycl
        IF (i < ncycl) THEN
          ip1 = i + 1
        ELSE
          ip1 = 1
        ENDIF
        WRITE(0,1001) cycl(i), modul(cycl(i)), ' USE ', cycl(ip1),
     >                modul(cycl(ip1))
 1001   FORMAT(I4,2X,A26,A5,I4,2X,A26)
      END DO
*
      END SUBROUTINE print_cycle



*=====================================================================
C IORDO                       TOUS HBUNG     06/03/11 20:06:16
!
!---------------- algorithme de TRI ---------------------------
!  Feb 2006
!     Avant on utilise pour le tri, l'algorithme de HOARE ecrit
!     en Fortran.
!
!     Ce algorithme est tres performant et le temps CPU ~ N*log(N)
!     Mais lorsque le tableau initial est DEJA ORDONNE
!     on a CPU ~ N*N (Cf. Numerical recipes in Fortran 77 : The Art
!     of scientific Computing (1992) Cambridge University Press)
```

```
!
!     On s'apercoit de ce grave inconvenient dans LIRLEC, avec le
!     maillage de grande taille (nelem > 1E6), lorsque on tente de
!     trier la liste dans l'ordre croissant
!     (ex pour 1 tableau de 1200000, iordo prend 1700 s CPU)
!
!     Nous cherchons un autre algorithme de tri, il y a l'algo de
!     HOARE, modifié par SINGLETON. Le Fortran est ecrit par JONES &
!     WISNIEWSKI  qui est performant quelque soit la configuration
!     initiale. Nousl'avons appelé IORDO_HSJW (les 4 initiales
!     Hoare, Singleton, Jones, Wisniewski).
!     Nous avons testé (sur Pentium 2GHz)

!     1) Tableau initial DEJA ORDONNE
!       Longueur du tableau     TCPU_HOARE(s)        TCPU_HSJW(s)
!            120000                 8.0                  0.00
!            570000               334.2                  0.05
!           1230000              1713.8                  0.11
!     2) Tableau initial ALEATOIRE
!       Longueur du tableau     TCPU_HOARE(s)        TCPU_HSJW(s)
!            120000                 0.02                 0.0
!            570000                 0.12                 0.14
!           1230000                 0.29                 0.33
!
!     Ce tableau montre que l'algorithme de Hoare est tres performant
!     mais le temps CPU croit fortement lors que le tableau est DEJA
!     ORDONNE.
!     Quant a L'algorithme HOARE-SINGLETON, il reste toujours ~ N*log(N),
!     quelque soit la nature du tableau initial.
!
!     Comme dans la plupart des cas dans Europlexus, le tableau initial
!     est DEJA ORDONNE (par exemple la lecture des listes d'elements ou
!     de noeud), la routine standard IORDO fait maintenant appel a
!     IORDO_HSJW.
!
!    Routines cotenues dans ce fichier :
!     1) IORDO_HSJW:  algorithme de HOARE modifie par SINGLRTON
!     1) IORDO_HOARE: algorithme de HOARE initial
!     2) IORDO_BUCKET:  utilise le tri par casier
!                                                    H. Bung
!
      SUBROUTINE IORDO(N,ITAB,IOP)
*
*     ----------------------------------------------------------------
*
*       tri rapide utilisant qsort dans l'ordre croissant ou decroissant
*                                                  h.  bung 04-88
*     ----------------------------------------------------------------
*
*             n        : longueur de itab
*             itab     : tableau a modifier
*          croissant
*             iop      : =0  les valeurs identiques sont consecutives
*                        =1  on vire les doubles ( n est modifie )
*          decroissant
*             iop      : =-2 les valeurs identiques sont consecutives
*                        =-1 on vire les doubles ( n est modifie )
*             si n .le. 1     : retour sans modification
*
*
*
*
      IMPLICIT NONE
*
*
      INTEGER, INTENT(INOUT) :: N,ITAB(*)
      INTEGER, INTENT(IN)    :: IOP
*
      INTEGER :: I, NN
*
*
*
      IF(N < 2) RETURN

      IF(IOP < 0) THEN
!---      tri decroissant: phase prelim. itab(i) = -itab(I)
         DO I=1,N
          ITAB(I) = - ITAB(I)
       END DO
      ENDIF

!!-----   Tri rapide : HOARE-SINGLETON
      CALL IORDO_HSJW(N, ITAB)
*
*
*-----   on elimine les doubles (si abs(iop)=1 )
      IF(IABS(IOP) == 1) THEN
         NN=1
         DO I=2,N
           IF(ITAB(I) == ITAB(NN)) CYCLE
           NN=NN+1
           IF(NN.NE.I) ITAB(NN)=ITAB(I)
         END DO
!-----       N est modifie
         N=NN
      ENDIF

      IF(IOP < 0) THEN
!---      tri decroissant: on remet itab(i) = -itab(I)
         DO I=1,N
          ITAB(I) = - ITAB(I)
       END DO
      ENDIF
*
      END SUBROUTINE IORDO

*=======================================================================

      SUBROUTINE IORDO_HSJW(N, IX)

!!  Tri Hoare-Singleton, Fortran ecrit par Jones & Wisniewski (SSORT)

****begin prologue  ssort
****date written   761101   (yymmdd)
****revision date  820801   (yymmdd)
****category no.  n6a2b1
****keywords  quicksort,singleton quicksort,sort,sorting
****author  jones, r. e., (snla)
*           wisniewski, j. a., (snla)
****purpose  ssort sorts array x and optionally makes the same
*            interchanges in array y.  the array x may be sorted in
*            increasing order or decreasing order.  a slightly modified
*            quicksort algorithm is used.
****description
*
*     written by rondall e. jones
*     modified by john a. wisniewski to use the singleton quicksort
*     algorithm.  date 18 november 1976.
*
*     abstract
*         ssort sorts array x and optionally makes the same
*         interchanges in array y.  the array x may be sorted in
*         increasing order or decreasing order.  a slightly modified
*         quicksort algorithm is used.
*
*     reference
*         singleton, r. c., algorithm 347, an efficient algorithm for
*         sorting with minimal storage, cacm,12(3),1969,185-7.
*
*     description of parameters
*         x - array of values to be sorted   (usually abscissas)
*         y - array to be (optionally) carried along
*         n - number of values in array x to be sorted
*         kflag - control parameter
*             =2  means sort x in increasing order and carry y along.
*             =1  means sort x in increasing order (ignoring y)
*             =-1 means sort x in decreasing order (ignoring y)
*             =-2 means sort x in decreasing order and carry y along.
****references  singleton,r.c., algorithm 347, an efficient algorithm
*                 for sorting with minimal storage, cacm,12(3),1969,
*                 185-7.
****routines called  xerror
****end prologue  ssort

      IMPLICIT NONE

!---      Dummy variables
      INTEGER, INTENT(IN) :: N
      INTEGER, INTENT(INOUT) :: IX(N)

!---       local variables
!!     M=DIM(IL)=DIM(IU)) : on peur traiter jusqu'a N < 2**M
      INTEGER :: IL(33),IU(33)
!!hb       INTEGER :: KFLAG=1     !  ordre croissant
      INTEGER :: NN, M, I, J, K, L, IJ
      REAL    ::  R
      INTEGER :: IT,KT


      NN = N
*
* sort x only
*
  100 CONTINUE
      M=1
      I=1
      J=NN
      R=.375
  110 IF (I .EQ. J) GO TO 155
  115 IF (R .GT. .5898437) GO TO 120
      R=R+3.90625E-2
      GO TO 125
  120 R=R-.21875
  125 K=I
*                                   select a central element of the
*                                   array and save it in location it
      IJ = I + (J-I) * R
      IT=IX(IJ)
*                                   if first element of array is greater
*                                   than it, interchange with it
      IF (IX(I) .LE. IT) GO TO 130
      IX(IJ)=IX(I)
      IX(I)=IT
      IT=IX(IJ)
  130 L=J
*                                   if last element of array is less than
*                                   it, interchange with it
      IF (IX(J) .GE. IT) GO TO 140
      IX(IJ)=IX(J)
      IX(J)=IT
      IT=IX(IJ)
*                                   if first element of array is greater
*                                   than it, interchange with it
      IF (IX(I) .LE. IT) GO TO 140
      IX(IJ)=IX(I)
      IX(I)=IT
      IT=IX(IJ)
      GO TO 140
  135 KT=IX(L)
      IX(L)=IX(K)
      IX(K)=KT
*                                   find an element in the second half of
*                                   the array which is smaller than it
  140 L=L-1
      IF (IX(L) .GT. IT) GO TO 140
*                                   find an element in the first half of
*                                   the array which is greater than it
  145 K=K+1
      IF (IX(K) .LT. IT) GO TO 145
*                                   interchange these elements
      IF (K .LE. L) GO TO 135
*                                   save upper and lower subscripts of
*                                   the array yet to be sorted
      IF (L-I .LE. J-K) GO TO 150
      IL(M)=I
      IU(M)=L
      I=K
      M=M+1
      GO TO 160
  150 IL(M)=K
      IU(M)=J
      J=L
      M=M+1
      GO TO 160
*                                   begin again on another portion of
*                                   the unsorted array
```

```
  155 M=M-1
      IF (M .EQ. 0) GO TO 300
      I=IL(M)
      J=IU(M)
  160 IF (J-I .GE. 1) GO TO 125
      IF (I .EQ. 1) GO TO 110
      I=I-1
  165 I=I+1
      IF (I .EQ. J) GO TO 155
      IT=IX(I+1)
      IF (IX(I) .LE. IT) GO TO 165
      K=I
  170 IX(K+1)=IX(K)
      K=K-1
      IF (IT .LT. IX(K)) GO TO 170
      IX(K+1)=IT
      GO TO 165
*
  300 CONTINUE

      END SUBROUTINE IORDO_HSJW

*=====================================================================

      SUBROUTINE IORDO_HOARE(N,ITAB,IOP)
*
*     ----------------------------------------------------------------
*
*       tri rapide (hoare) ordre croissant ou decroissant
*                                                  h.  bung 04-88
*     ----------------------------------------------------------------
*
*             n        : longueur de itab
*             itab     : tableau a modifier
*          croissant
*             iop      : =0  les valeurs identiques sont consecutives
*                        =1  on vire les doubles ( n est modifie )
*          decroissant
*             iop      : =-2 les valeurs identiques sont consecutives
*                        =-1 on vire les doubles ( n est modifie )
*             si n .le. 1     : retour sans modification
*
*
*
*
      IMPLICIT NONE
*
*
      INTEGER, INTENT(INOUT) :: N,ITAB(*)
      INTEGER, INTENT(IN)    :: IOP
*
      INTEGER :: I,NN,IR,IRA,IAUX,IRB,IQ,IA,IB,J,LPIVOT
*
*-----    variables locales pour les piles de doublets
*
      INTEGER, PARAMETER :: LGPILE=100
      INTEGER :: KOPF, IPIL1(LGPILE),IPIL2(LGPILE)
*
*
      IF(N < 2) RETURN
*
      IPIL1(1)=1        !AVANT: CALL PILINI & CALL PILALL(1,N))
      IPIL2(1)=N
      KOPF=1

      I=0
      J=0
*
  1   CONTINUE
!!!avant :       CALL PILFRE(IA,IB)
      IA=IPIL1(KOPF)
      IB=IPIL2(KOPF)
      KOPF=KOPF-1
  2   IF(IA.LT.IB) THEN
         LPIVOT=ITAB(IA)
*--          exploration de ia+1 a ib
         IQ=IA+1
         IR=IB
   3     CONTINUE
         IF(IQ.LE.IR) THEN
             J=IR-1
             DO I=IQ,J
                IF(ITAB(I).GT.LPIVOT) GOTO 5
             END DO
   5         IQ=I+1
             DO J=IR,I+1,-1
                IF(ITAB(J).LT.LPIVOT) GOTO 7
             END DO
   7         CONTINUE
***avant:    call iswap(itab(i),itab(j))
             IAUX=ITAB(I)
             ITAB(I)=ITAB(J)
             ITAB(J)=IAUX

             IR=J-1
             GOTO 3
         ENDIF
*---         exploration terminee , on a 2 partitions a trier
         IF(I.EQ.J .AND. ITAB(J).LT.LPIVOT) IR=J
*--          on met le pivot a sa place definitive
***avant:         call iswap(itab(ia),itab(ir))
         IAUX=ITAB(IA)
         ITAB(IA)=ITAB(IR)
         ITAB(IR)=IAUX
*---         on empile la partition la +grande,si elle contient 2 elts
         IRA=IR-1
         IRB=IR+1
         IF(IR-IA .GT. IB-IR) THEN
             IF(IA.LT.IRA) THEN
***avant:                call pilall(ia,ira)
                 IF(KOPF >= LGPILE ) THEN
                    STOP 'IORDO :PILES SATURéES 2'
                 ENDIF
                 KOPF=KOPF+1
                 IPIL1(KOPF)=IA
                 IPIL2(KOPF)=IRA
             ENDIF
             IA=IRB
          ELSE
             IF(IRB.LT.IB) THEN
***avant:                 call pilall(irb,ib)
                 IF(KOPF >= LGPILE ) THEN
                    STOP 'IORDO :PILES SATURéES 2'
                 ENDIF
                 KOPF=KOPF+1
                 IPIL1(KOPF)=IRB
                 IPIL2(KOPF)=IB
             ENDIF
             IB=IRA
          ENDIF
          GOTO 2
       ENDIF

       IF(KOPF > 0 ) GOTO 1

*
*-----   le tri est decroissant
       IF(IOP.LT.0) THEN
          NN=N/2
          J=N
            DO I=1,NN
            IAUX=ITAB(I)
            ITAB(I)=ITAB(J)
            ITAB(J)=IAUX
            J=J-1
          END DO
       ENDIF
*
*-----   on elimine les doubles (si abs(iop)=1 )
       IF(IABS(IOP).EQ.1) THEN
          NN=1
          DO I=2,N
            IF(ITAB(I).EQ.ITAB(NN)) CYCLE
            NN=NN+1
            IF(NN.NE.I) ITAB(NN)=ITAB(I)
          END DO
*-----       n est modifie
          N=NN
       ENDIF
*
       END SUBROUTINE IORDO_HOARE

*=====================================================================

       SUBROUTINE IORDO_BUCKET(N,ITAB,IOP)
*
*     ----------------------------------------------------------------
*
*     tri rapide utilisant le tri casier ordre croissant ou decroissant
*
*                                                  p.  galon 03-2006
*     ----------------------------------------------------------------
*
*             n        : longueur de itab
*             itab     : tableau a modifier
*          croissant
*             iop      : =0  les valeurs identiques sont consecutives
*                        =1  on vire les doubles ( n est modifie )
*          decroissant
*             iop      : =-2 les valeurs identiques sont consecutives
*                        =-1 on vire les doubles ( n est modifie )
*             si n .le. 1     : retour sans modification
*
* remarque : cet algorithme est performant pour n grand, disons > 100000
* et lorsque le nombre d'entiers entre le min et le max de itab est
* inferieur à nln n. la complexiteé de l'algorithme est toujours de l'or
* de o(n+m) ou m est le cardinal de ntrav (max -min + 1)
*
       IMPLICIT NONE
*
*
       INTEGER, INTENT(INOUT) :: N,ITAB(*)
       INTEGER, INTENT(IN)    :: IOP
*
       INTEGER :: I, NN,IAUX,  LONG, MIN, MAX, J, ICONT
       INTEGER , ALLOCATABLE :: NTRAV(:)
*
*
       IF(N < 2) RETURN
*
* --- valeurs mini et maxi du tableau et longueur
*
       MIN = MINVAL(ITAB(1:N))
       MAX = MAXVAL(ITAB(1:N))
       LONG = MAX - MIN + 1
*
       ALLOCATE (NTRAV(LONG))
*
* --- initialisation du tableau
*
       NTRAV(:) = 0
*
* --- on compte les occurences de chaque entier
*
       DO I = 1, N
          J = ITAB(I) - MIN + 1
          NTRAV(J) = NTRAV(J) + 1
       ENDDO
*
* --- on reconstruit le tableau initial
*     (dans l'ordre croissant)
*
       ICONT = 0
       DO I = 1, LONG
*
         DO WHILE (NTRAV(I) > 0)
           ICONT = ICONT + 1
           ITAB(ICONT) = MIN + I - 1
           NTRAV(I) = NTRAV(I) - 1
         END DO
*
       END DO
       IF( N /= ICONT) THEN
          STOP 'IORDO_BUCKET ERROR 1'
       ENDIF
*
```

```
      DEALLOCATE (NTRAV)
*
*-----   le tri est decroissant
*
      IF(IOP < 0) THEN
         NN=N/2
         J=N
           DO I=1,NN
           IAUX=ITAB(I)
           ITAB(I)=ITAB(J)
           ITAB(J)=IAUX
           J=J-1
         END DO
      ENDIF
*
*
*-----   on elimine les doubles (si abs(iop)=1 )
*
      IF(IABS(IOP).EQ.1) THEN
         NN=1
         DO I=2,N
           IF(ITAB(I) == ITAB(NN)) CYCLE
           NN=NN+1
           IF(NN.NE.I) ITAB(NN)=ITAB(I)
         END DO
!-----       N est modifie
         N=NN
      ENDIF
*
      END SUBROUTINE IORDO_BUCKET
```

reg.pl

```
use Win32API::Registry 0.13 qw( :ALL );

$sSubKey = "SOFTWARE\\EUROPLEXUS";

RegOpenKeyEx( HKEY_LOCAL_MACHINE, $sSubKey, 0, KEY_READ, $phKey) ||
  die "could not open the key\n";

RegQueryValueEx($phKey, 'Eurodir', [], $type, $euro, []);
RegQueryValueEx($phKey, 'Utildir', [], $type, $util, []);
#RegQueryValueEx($phKey, 'Folcdir', [], $type, $folc, []) ||
#  die "could not find Folcdir\n";

printf "EURO Directory is: $euro\n";
printf "UTIL Directory is: $util\n";
printf "FOLC Directory is: $folc\n";

exit;
```

sedall.pl

```
#
# Perform sedfile command with local sed_cmd on all given files

# All arguments are file names: do the globbing

@FileNames = glob(join($",@ARGV));

foreach $file (@FileNames) {
    print("Editing file $file\n");
    system("sedfile $file");
}
```

sedfile.pl

```
$name = $ARGV[0];
system("sed -f sed_cmd $name > _sed.txt");
system("cp -f _sed.txt $name");
system("rm -f _sed.txt");
```

stat_anom.pl

```
#-------------------------------------------------------------------------------
# Build up summary of OPEN "fiche d'anomalie" files in current directory
#-------------------------------------------------------------------------------
# Zero or one arguments are accepted
#
if ($#ARGV < -1 || $#ARGV > 0) {         # Index of last argument must be 0 or 1
  die "Usage: stat_anom [-f]\007\n";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#       -f       list only firts line of description
#
if ($#ARGV >= 0) { # There is at least one argument
  while ($ARGV[0] =~ /^-/) {
    $_ = shift;
    if (/^-f(.*)/) {
      $first = "yes";
    }
    else {
      die "Usage: stat_anom [-f]\007\n";
    }
  }
}
#-------------------------------------------------------------------------------
print("Resume des Fiches d'Anomalie\n");
print("============================\n\n");
print("Fiches Ouvertes:\n");
print("---------------\n");
print("No.  Ouverte     Par       Description\n");
print("-------------------------------------------------------------------\n");
#-------------------------------------------------------------------------------
@FileNames = glob("a*_o.txt");
#-------------------------------------------------------------------------------
foreach $file (@FileNames) {
#-------------------------------------------------------------------------------
# Verify that the file exists in the local directory
#
if ( ! -r $file ) {
  die "File $file does not exist in local directory!\007\n";
}
#-------------------------------------------------------------------------------
# Get fiche number
$num = $file;
$num =~ s/^a//;
$num =~ s/_o\.txt//;
# Get fiche opening date
$date = `grep -m 1 "^    Date            : " $file`;
$date =~ s/^    Date            : //;
$date =~ s/\n$//;
$date =~ s/ $//;
($weekd,$month,$day,$hour,$year) = split(/[ \t]+/, $date);
$len = length ($day);
if ($len == 1) {
  $day = " $day";
}
$date = "$day\_$month\_$year";
# Get author name
$aut = `grep -m 1 "^    Auteur          : " $file`;
$aut =~ s/^    Auteur          : //;
$aut =~ s/\n$//;
# Pad author to 9 characters with trailing blanks if necessary
$len = length ($aut);
if ($len == 0) {
  $aut = "         ";
}
elsif ($len == 1) {
  $aut = "$aut        ";
}
elsif ($len == 2) {
  $aut = "$aut       ";
}
elsif ($len == 3) {
  $aut = "$aut      ";
}
elsif ($len == 4) {
  $aut = "$aut     ";
}
elsif ($len == 5) {
  $aut = "$aut    ";
}
elsif ($len == 6) {
  $aut = "$aut   ";
}
elsif ($len == 7) {
  $aut = "$aut  ";
}
elsif ($len == 8) {
  $aut = "$aut ";
}
# Get description
undef $desc;
open (FICHE, $file);
while (<FICHE>) {
  if ( m/^----- Begin Description du probleme :/ ) {
    $echo = 1;
  }
  elsif ( m/^----- End Description du probleme :/ ) {
    undef $echo;
    close FICHE;
  }
  else {
    if (defined ($echo)) {
      if ( m/^[ \t]*\n/ ) {
        #print "Blank line read.\n";
      }
      else {
        $des = $_;
        $des =~ s/\n$//;
        if (defined ($first)) {
          $desc = $des;
          undef $echo;
          close FICHE;
        }
        else {
          if (defined ($desc)) {
            $desc = "$desc\n                              $des";
          }
          else {
            $desc = $des;
          }
        }
      }
    }
  }
  #
}
close FICHE;
#
  print("$num $date $aut $desc\n");
#
}
#-------------------------------------------------------------------------------
print("\n\n\n");
print("Fiches Fermees:\n");
print("--------------\n");
print("No.  Ouverte     Par       Fermee   Par       Description\n");
print("-------------------------------------------------------------------\n");
#-------------------------------------------------------------------------------
@FileNames = glob("a*_f.txt");
#-------------------------------------------------------------------------------
foreach $file (@FileNames) {
  #-----------------------------------------------------------------------------
  # Verify that the file exists in the local directory
  #
  if ( ! -r $file ) {
    die "File $file does not exist in local directory!\007\n";
  }
  #-----------------------------------------------------------------------------
  # Get fiche number
  $num = $file;
  $num =~ s/^a//;
  $num =~ s/_f\.txt//;
```

```
  # Get fiche opening date
  $date = `grep -m 1 "^    Date            : " $file`;
  $date =~ s/^    Date            : //;
  $date =~ s/\n$//;
  $date =~ s/ $//;
  ($weekd,$month,$day,$hour,$year) = split(/[ \t]+/, $date);
  $len = length ($day);
  if ($len == 1) {
    $day = " $day";
  }
  $date = "$day\_$month\_$year";
  # Get author name
  $aut = `grep -m 1 "^    Auteur          : " $file`;
  $aut =~ s/^    Auteur          : //;
  $aut =~ s/\n$//;
  # Pad author to 9 characters with trailing blanks if necessary
  $len = length ($aut);
  if ($len == 0) {
    $aut = "         ";
  }
  elsif ($len == 1) {
    $aut = "$aut        ";
  }
  elsif ($len == 2) {
    $aut = "$aut       ";
  }
  elsif ($len == 3) {
    $aut = "$aut      ";
  }
  elsif ($len == 4) {
    $aut = "$aut     ";
  }
  elsif ($len == 5) {
    $aut = "$aut    ";
  }
  elsif ($len == 6) {
    $aut = "$aut   ";
  }
  elsif ($len == 7) {
    $aut = "$aut  ";
  }
  elsif ($len == 8) {
    $aut = "$aut ";
  }
  # Get fiche closure date and closing author name
  $line = `grep -m 1 "EST FERMEE PAR" $file`;
  #print "Line:$line\n";
  ($f,$n,$nn,$e,$f1,$p,$autc,$l,$datec) = split(/[ \t]+/, $line);
  $autc =~ s/''//g;
  $autc =~ s/\n$//;
  $datec =~ s/\n$//;
  # Pad author to 9 characters with trailing blanks if necessary
  $len = length ($autc);
  if ($len == 0) {
    $autc = "         ";
  }
  elsif ($len == 1) {
    $autc = "$autc        ";
  }
  elsif ($len == 2) {
    $autc = "$autc       ";
  }
  elsif ($len == 3) {
    $autc = "$autc      ";
  }
  elsif ($len == 4) {
    $autc = "$autc     ";
  }
  elsif ($len == 5) {
    $autc = "$autc    ";
  }
  elsif ($len == 6) {
    $autc = "$autc   ";
  }
  elsif ($len == 7) {
    $autc = "$autc  ";
  }
  elsif ($len == 8) {
    $autc = "$autc ";
  }
  # Get description
  undef $desc;
  open (FICHE, $file);
  while (<FICHE>) {
    if ( m/^----- Begin Description du probleme :/ ) {
      $echo = 1;
    }
    elsif ( m/^----- End Description du probleme :/ ) {
      undef $echo;
      close FICHE;
    }
    else {
      if (defined ($echo)) {
        if ( m/^[ \t]*\n/ ) {
          #print "Blank line read.\n";
        }
        else {
          $des = $_;
          $des =~ s/\n$//;
          if (defined ($first)) {
            $desc = $des;
            undef $echo;
            close FICHE;
          }
          else {
            if (defined ($desc)) {
              $desc = "$desc\n                                                $des
";
            }
            else {
              $desc = $des;
            }
          }
        }
      }
    }
    #
  }
  close FICHE;
#
  print("$num $date $aut $datec $autc $desc\n");
#
}
exit;
```

stat_deve.pl

```
#-------------------------------------------------------------------------------
# Build up summary of OPEN "fiche de developpement" files in current directory
#-------------------------------------------------------------------------------
# Zero or one arguments are accepted
#
if ($#ARGV < -1 || $#ARGV > 0) {          # Index of last argument must be 0 or 1
  die "Usage: stat_deve [-f]\007\n";
}
#-------------------------------------------------------------------------------
# Process optional switches:
#       -f       list only firts line of description
#
if ($#ARGV >= 0) { # There is at least one argument
  while ($ARGV[0] =~ /^-/) {
    $_ = shift;
    if (/^-f(.*)/) {
      $first = "yes";
    }
    else {
      die "Usage: stat_deve [-f]\007\n";
    }
  }
}
#-------------------------------------------------------------------------------
print("Resume des Fiches de Developpement\n");
print("==================================\n\n");
print("Fiches Ouvertes:\n");
print("----------------\n");
print("No.  Ouverte     Par       Titre\n");
print("-------------------------------------------------------------------------\n");
#-------------------------------------------------------------------------------
@FileNames = glob("d*_o.txt");
#-------------------------------------------------------------------------------
foreach $file (@FileNames) {
  #-----------------------------------------------------------------------------
  # Verify that the file exists in the local directory
  #
  if ( ! -r $file ) {
    die "File $file does not exist in local directory!\007\n";
  }
  #-----------------------------------------------------------------------------
  # Get fiche number
  $num = $file;
  $num =~ s/^d//;
  $num =~ s/_o\.txt//;
  # Get fiche opening date
  $date = `grep -m 1 "^    Date            : " $file`;
  $date =~ s/^    Date            : //;
  $date =~ s/\n$//;
  $date =~ s/ $//;
  ($weekd,$month,$day,$hour,$year) = split(/[ \t]+/, $date);
  $len = length ($day);
  if ($len == 1) {
    $day = " $day";
  }
  $date = "$day\_$month\_$year";
  # Get author name
  $aut = `grep -m 1 "^    Auteur          : " $file`;
  $aut =~ s/^    Auteur          : //;
  $aut =~ s/\n$//;
  # Pad author to 9 characters with trailing blanks if necessary
  $len = length ($aut);
  if ($len == 0) {
    $aut = "         ";
  }
  elsif ($len == 1) {
    $aut = "$aut        ";
  }
  elsif ($len == 2) {
    $aut = "$aut       ";
  }
  elsif ($len == 3) {
    $aut = "$aut      ";
  }
  elsif ($len == 4) {
    $aut = "$aut     ";
  }
  elsif ($len == 5) {
    $aut = "$aut    ";
  }
  elsif ($len == 6) {
    $aut = "$aut   ";
  }
  elsif ($len == 7) {
    $aut = "$aut  ";
  }
  elsif ($len == 8) {
    $aut = "$aut ";
  }
  # Get description
  undef $desc;
  open (FICHE, $file);
  while (<FICHE>) {
    if ( m/^----- Begin Titre  :/ ) {
      $echo = 1;
    }
    elsif ( m/^----- End Titre/ ) {
      undef $echo;
      close FICHE;
    }
    else {
      if (defined ($echo)) {
        if ( m/^[ \t]*\n/ ) {
          #print "Blank line read.\n";
        }
        else {
          $des = $_;
```

```
          $des =~ s/\n$//;
          if (defined ($first)) {
            $desc = $des;
            undef $echo;
            close FICHE;
          }
          else {
            if (defined ($desc)) {
              $desc = "$desc\n                          $des";
            }
            else {
              $desc = $des;
            }
          }
        }
      }
    }
    #
  }
  close FICHE;
#
  print("$num $date $aut $desc\n");
#
}
#------------------------------------------------------------------------------
print("\n\n\n");
print("Fiches Fermees:\n");
print("--------------\n");
print("No.  Ouverte     Par       Fermee   Par       Titre\n");
print("------------------------------------------------------------------------
---\n");
#------------------------------------------------------------------------------
@FileNames = glob("d*_f.txt");
#------------------------------------------------------------------------------
foreach $file (@FileNames) {
  #----------------------------------------------------------------------------
  # Verify that the file exists in the local directory
  #
  if ( ! -r $file ) {
    die "File $file does not exist in local directory!\007\n";
  }
  #----------------------------------------------------------------------------
  # Get fiche number
  $num = $file;
  $num =~ s/^d//;
  $num =~ s/_f\.txt//;
  # Get fiche opening date
  $date = `grep -m 1 "^    Date            : " $file`;
  $date =~ s/^    Date            : //;
  $date =~ s/\n$//;
  $date =~ s/ $//;
  ($weekd,$month,$day,$hour,$year) = split(/[ \t]+/, $date);
  $len = length ($day);
  if ($len == 1) {
    $day = " $day";
  }
  $date = "$day\_$month\_$year";
  # Get author name
  $aut = `grep -m 1 "^    Auteur          : " $file`;
  $aut =~ s/^    Auteur          : //;
  $aut =~ s/\n$//;
  # Pad author to 9 characters with trailing blanks if necessary
  $len = length ($aut);
  if ($len == 0) {
    $aut = "         ";
  }
  elsif ($len == 1) {
    $aut = "$aut        ";
  }
  elsif ($len == 2) {
    $aut = "$aut       ";
  }
  elsif ($len == 3) {
    $aut = "$aut      ";
  }
  elsif ($len == 4) {
    $aut = "$aut     ";
  }
  elsif ($len == 5) {
    $aut = "$aut    ";
  }
  elsif ($len == 6) {
    $aut = "$aut   ";
  }
  elsif ($len == 7) {
    $aut = "$aut  ";
  }
  elsif ($len == 8) {
    $aut = "$aut ";
  }
  # Get fiche closure date and closing author name
  $line = `grep -m 1 "EST FERMEE PAR" $file`;
  #print "Line:$line\n";
  ($f,$n,$nn,$e,$f1,$p,$autc,$l,$datec) = split(/[ \t]+/, $line);
  $autc =~ s/'//g;
  $autc =~ s/\n$//;
  $datec =~ s/\n$//;
  # Pad author to 9 characters with trailing blanks if necessary
  $len = length ($autc);
  if ($len == 0) {
    $autc = "         ";
  }
  elsif ($len == 1) {
    $autc = "$autc        ";
  }
  elsif ($len == 2) {
    $autc = "$autc       ";
  }
  elsif ($len == 3) {
    $autc = "$autc      ";
  }
  elsif ($len == 4) {
    $autc = "$autc     ";
  }
  elsif ($len == 5) {
    $autc = "$autc    ";
  }
  elsif ($len == 6) {
    $autc = "$autc   ";
  }
  elsif ($len == 7) {
    $autc = "$autc  ";
  }
  elsif ($len == 8) {
    $autc = "$autc ";
  }
  # Get description
  undef $desc;
  open (FICHE, $file);
  while (<FICHE>) {
    if ( m/^----- Begin Titre  :/ ) {
      $echo = 1;
    }
    elsif ( m/^----- End Titre/ ) {
      undef $echo;
      close FICHE;
    }
    else {
      if (defined ($echo)) {
        if ( m/^[ \t]*\n/ ) {
          #print "Blank line read.\n";
        }
        else {
          $des = $_;
          $des =~ s/\n$//;
          if (defined ($first)) {
            $desc = $des;
            undef $echo;
            close FICHE;
          }
          else {
            if (defined ($desc)) {
              $desc = "$desc\n                                                  $des
";
            }
            else {
              $desc = $des;
            }
          }
        }
      }
    }
    #
  }
  close FICHE;
#
  print("$num $date $aut $datec $autc $desc\n");
#
}
exit;
```

test.txt

```
This is just a test, please ignore it!
```

testat1.pl

```
#------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#------------------------------------------------------------------------------
# Check if the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  die "The EUROPLEXUS environment variable is undefined!\007\n";
}
#------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  die "The EUROPLEXUS directory: $epx does not exist!\007\n";
}
chdir "$epx\\Fromcentral" || die "Can't chdir to Fromcentral!\n";
system("epx_get celem");
exit;
```

testat2.pl

```
#------------------------------------------------------------------------------
# Make STDOUT and STDERR unbuffered
#
$oldfh = select(STDOUT); $| = 1; select($oldfh);
$oldfh = select(STDERR); $| = 1; select($oldfh);
#------------------------------------------------------------------------------
# Check if the EUROPLEXUS variable is set
#
$epx = $ENV{'EUROPLEXUS'};
if ( ! defined $epx) {
  die "The EUROPLEXUS environment variable is undefined!\007\n";
}
#------------------------------------------------------------------------------
# Check that the EUROPLEXUS directory exists
#
if ( ! -d $epx) {
  die "The EUROPLEXUS directory: $epx does not exist!\007\n";
}
chdir "$epx\\Fromcentral\\Tmp" || die "Can't chdir to Fromcentral\\Tmp!\n";
system("epx_get celem");
system("epx_cmp -o");
system("epx_lk -w");
exit;
```

European Commission
EUR 25821 EN – Joint Research Centre – Institute for the Protection and Security of the Citizen

Title: Organisation of the EUROPLEXUS Mirror Site (MS-Windows) at JRC Ispra Seventh Edition

Author(s): F. Casadei, G. Solomos, G. Valsamos, H. Bung, A. Beccantini

Luxembourg: Publications Office of the European Union

2013 – 306 pp. – 21.0 x 29.7 cm

EUR – Scientific and Technical Research series – ISSN 1831-9424

ISBN 978-92-79-28746-6

doi: 10.2788/8488

Abstract

This document describes the organisation of the "mirror site" for the development of the EUROPLEXUS computer code at the Joint Research Centre (JRC) of the European Commission (EC) at Ispra. It reflects the status of the mirror site and its interaction features with the CEA site as of February 2013. The main principles of EUROPLEXUS project and the general architecture of multi-site development are outlined.

LB-NA-25821-EN-N

As the Commission's in-house science service, the Joint Research Centre's mission is to provide EU policies with independent, evidence-based scientific and technical support throughout the whole policy cycle.

Working in close cooperation with policy Directorates-General, the JRC addresses key societal challenges while stimulating innovation through developing new standards, methods and tools, and sharing and transferring its know-how to the Member States and international community.

Key policy areas include: environment and climate change; energy and transport; agriculture and food security; health and consumer protection; information society and digital agenda; safety and security including nuclear; all supported through a cross-cutting and multi-disciplinary approach.

Publications Office

ISBN 978-92-79-28746-6