Installation guide for *esys-Escript*

Release - 5.3
(r6774)

Escript development team

January 12, 2019

Centre for Geoscience Computing (GeoComp)
The University of Queensland
Brisbane, Australia
Email: esys@esscc.uq.edu.au

Guide to Documentation

Documentation for `esys.escript` comes in a number of parts. Here is a rough guide to what goes where.

---

install.pdf	"Installation guide for *esys-Escript*": Instructions for compiling *escript* for your system from its source code.
cookbook.pdf	"The *escript* COOKBOOK": An introduction to *escript* for new users from a geophysics perspective.
user.pdf	"*esys-Escript* User's Guide: Solving Partial Differential Equations with Escript and Finley": Covers main *escript* concepts.
inversion.pdf	"`esys.downunder`: Inversion with *escript*": Explanation of the inversion toolbox for *escript*.
sphinx_api directory	Documentation for *escript Python* libraries.
escript_examples(.tar.gz)/(.zip)	Full example scripts referred to by other parts of the documentation.
doxygen directory	Documentation for C++ libraries (mostly only of interest for developers).

Contents

CHAPTER
ONE

Introduction

This document describes how to install *esys-Escript*[1] on to your computer. To learn how to use Escript please see the Cookbook, User's guide or the API documentation.

Escript is primarily developed on Linux desktop, SGI ICE and MacOS X systems. It can be installed in two ways:

1. Binary packages – ready to run with no compilation required. These will hopefully be available in upcomming Debian and Ubuntu releases so just use your normal package manager (so you don't need this guide).
2. From source – that is, it must be compiled for your machine. This is the topic of this guide.

See the site `https://answers.launchpad.net/escript-finley` for online help. Chapter 2 covers installing from source. Appendix A lists some c++ features which your compiler must support in order to compile escript. This version of escript has the option of using `Trilinos` in addition to our regular solvers. Appendix B covers features of `Trilinos` which escript needs.

[1] For the rest of the document we will drop the *esys-*

CHAPTER
TWO

Installing from Source

This chapter assumes you are using a unix/posix like system (including MacOSX).

2.1 Parallel Technologies

It is likely that the computer you run `esys.escript` on, will have more than one processor core. Escript can make use of multiple cores [in order to solve problems more quickly] if it is told to do so, but this functionality must be enabled at compile time. Section 2.1.1 gives some rough guidelines to help you determine what you need.

There are two technologies which `esys.escript` can employ here.

- OpenMP – the more efficient of the two [thread level parallelism].
- MPI – Uses multiple processes (less efficient), needs less help from the compiler.

Escript is primarily tested on recent versions of the GNU and Intel suites ("g++" / "icpc"). However, it also passes our tests when compiled using "clang++". Escript now requires compiler support for some features of the C++11 standard. See Appendix A for a list.

Our current test compilers include:

- g++ 6
- clang++ 7
- intel icpc v17

Note that:

- OpenMP will not function correctly for g++ $\leq$ 4.2.1 (and is not currently supported by clang).
- icpc v11 has a subtle bug involving OpenMP and C++ exception handling, so this combination should not be used.

2.1.1 What parallel technology do I need?

If you are using any version of Linux released in the past few years, then your system compiler will support OpenMP with no extra work (and give better performance); so you should use it. You will not need MPI unless your computer is some form of cluster.

If you are using BSD or MacOSX and you are just experimenting with `esys.escript`, then performance is probably not a major issue for you at the moment so you don't need to use either OpenMP or MPI. This also applies if you write and polish your scripts on your computer and then send them to a cluster to execute. If in the future you find escript useful and your scripts take significant time to run, then you may want to recompile `esys.escript` with more options.

Note that even if your version of esys.escript has support for OpenMP or MPI, you will still need to tell the system to use it when you run your scripts. If you are using the run-escript launcher, then this is controlled by the -t, -p, and -n options. If not, then consult the documentation for your MPI libraries (or the compiler documentation in the case of OpenMP [1]).

If you are using MacOSX, then see the next section, if not, then skip to Section 2.3.

2.2 MacOS

This release of esys.escript has only been tested on OSX 10.13. For this section we assume you are using either homebrew or MacPorts as a package manager[2]. You can of course install prerequisite software in other ways. For example, we have had *some* success changing the default compilers used by those systems. However this is more complicated and we do not provide a guide here.
Both of those systems require the XCode command line tools to be installed[3].

2.3 Building

To simplify things for people, we have prepared _options.py files for a number of systems[4]. The options files are located in the scons/templates directory. We suggest that the file most relevant to your OS be copied from the templates directory to the scons directory and renamed to the form XXXX_options.py where XXXX should be replaced with your computer's (host-)name. If your particular system is not in the list below, or if you want a more customised build, see Section 2.3.9 for instructions.

- Debian - 2.3.1
- Ubuntu - 2.3.2
- OpenSuse - 2.3.3
- Centos - 2.3.4
- Fedora - 2.3.5
- MacOS (macports) - 2.3.6
- MacOS (homebrew) - 2.3.7
- FreeBSD - 2.3.8

Once these are done proceed to Section 2.4 for cleanup steps.

All of these instructions assume that you have obtained the esys.escript source (and uncompressed it if necessary).

2.3.1 Debian

```
sudo aptitude install python-dev python-numpy libboost-python-dev libnetcdf-dev
sudo aptitude install libnetcdf-cxx-legacy-dev libnetcdf-c++4-dev
sudo aptitude install scons lsb-release libboost-random-dev
sudo aptitude install python-sympy python-matplotlib python-scipy
sudo aptitude install python-pyproj python-gdal
```

If you are running *Wheezy*, you can use:

```
sudo aptitude install gmsh
```

[1] It may be enough to set the OMP_NUM_THREADS environment variable.

[2] Note that package managers will make changes to your computer based on programs configured by other people from various places around the internet. It is important to satisfy yourself as to the security of those systems.

[3] As of OSX10.9, the command xcode-select --install will allow you to download and install the commandline tools.

[4] These are correct a time of writing but later versions of those systems may require tweaks. Also, these systems represent a cross section of possible platforms rather than meaning those systems get particular support.

to add extra meshing functionality.

Optional step
If for some reason, you wish to rebuild the documentation, you would also need the following:

```
sudo aptitude install python-sphinx doxygen python-docutils texlive
sudo aptitude install zip texlive-latex-extra latex-xcolor
```

In the source directory execute the following (substitute jessie for XXXX):

```
scons -j1 options_file=scons/templates/XXXX_options.py
```

If you wish to test your build, you can use the following:

```
scons -j1 py_tests options_file=scons/templates/XXXX_options.py
```

2.3.2 Ubuntu

If you have not installed `aptitude`, then substitute `apt-get` in the following.

```
sudo aptitude install python-dev python-numpy libboost-python-dev
sudo aptitude install libnetcdf-cxx-legacy-dev libnetcdf-c++4-dev
sudo aptitude install libnetcdf-dev libboost-random-dev
sudo aptitude install scons lsb-release
sudo aptitude install python-sympy python-matplotlib python-scipy
sudo aptitude install python-pyproj python-gdal gmsh
```

Optional step
If for some reason, you wish to rebuild the documentation, you would also need the following:

```
sudo aptitude install python-sphinx doxygen python-docutils texlive
sudo aptitude install zip texlive-latex-extra latex-xcolor
```

In the source directory execute the following (substitute precise, quantal or raring as appropriate for XXXX):

```
scons -j1 options_file=scons/templates/XXXX_options.py
```

If you wish to test your build, you can use the following:

```
scons -j1 py_tests options_file=scons/templates/XXXX_options.py
```

2.3.3 OpenSuse

These instructions were prepared using openSUSE version 15.0.
Install packages from the main distribution:

```
sudo zypper in python-devel python2-numpy python-gdal
sudo zypper in python2-scipy python2-sympy python2-matplotlib
sudo zypper in gcc gcc-c++ scons netcdf-devel libnetcdf_c++-devel
sudo zypper in libboost_python-py2_7-1_66_0-devel libboost_numpy-py2_7-1_66_0-devel
sudo zypper in libboost_iostreams1_66_0-devel suitesparse-devel
```

Now to build escript itself. In the escript source directory:

```
scons -j1 options_file=scons/templates/opensuse_options.py
```

If you wish to test your build, you can use the following:

```
scons -j1 py_tests options_file=scons/templates/opensuse_options.py
```

Now go to Section 2.4 for cleanup.

2.3.4 CentOS

These instructions were prepared using CentOS release 7 (from late 2016). The core of escript works, however some functionality is not available because the default packages for some dependencies in CentOS are too old.
Add the `EPEL` repository.

```
yum install epel-release.noarch
```

Install packages:

```
yum install netcdf-devel netcdf-cxx-devel gdal-python
yum install python-devel numpy scipy sympy python2-scons
yum install python-matplotlib gcc gcc-c++ boost-devel
yum install boost-python gdal-devel suitesparse-devel
```

The above packages will allow you to use most features except the `esys.downunder` inversion library. If you wish to use those it, you will need to install some additional packages.
For some coordinate transformations, `esys.downunder` can also make use of the python interface to a tool called `proj`. There does not seem to be an obvious CentOS repository for this though. If it turns out to be necessary for your particular application, the source can be downloaded.
Now to build escript itself. In the escript source directory:

```
scons -j1 options_file=scons/templates/centos7_0_options.py
```

Now go to Section 2.4 for cleanup.

2.3.5 Fedora

These instructions were prepared using the 28 alpha.
Install packages

```
yum install netcdf-cxx-devel gcc-c++ scipy
yum install sympy scons pyproj gdal python-matplotlib
yum install boost-devel suitesparse-devel gmsh python2-gdal
```

Now to build escript itself. In the escript source directory:

```
scons -j1 options_file=scons/templates/fedora_options.py
```

If you wish to test your build, you can use the following:

```
scons -j1 py_tests options_file=scons/templates/fedora_options.py
```

Now go to Section 2.4 for cleanup.

2.3.6 MacOS 10.10 and later (macports)

The following will install the capabilities needed for the `macports_10.10_options.py` file (later versions can use the same options file).

```
sudo port install scons
sudo port select --set python python27
sudo port install boost
sudo port install py27-numpy
sudo port install py27-sympy
sudo port select --set py-sympy py27-sympy
sudo port install py27-scipy
sudo port install py27-pyproj
sudo port install py27-gdal
sudo port install netcdf-cxx
sudo port install silo
```

```
scons -j1 options_file=scons/templates/macports_10.10options.py
```

2.3.7 MacOS 10.13 and later (homebrew)

The following will install the capabilities needed for the `homebrew_10.13_options.py` file.

```
brew install scons
brew install boost-python
brew install netcdf
```

There do not appear to be formulae for `sympy` or `pyproj` so if you wish to use those features, then you will need to install them separately.

```
scons -j1 options_file=scons/templates/homebrew_10.13_options.py
```

2.3.8 FreeBSD

At time of writing, `numpy` does not install correctly on FreeBSD. Since `numpy` is a critical dependency for `esys.escript`, we have been unable to test on FreeBSD.

2.3.9 Other Systems / Custom Builds

`esys.escript` has support for a number of optional packages. Some, like `netcdf` need to be enabled at compile time, while others, such as `sympy` and the projection packages used in `esys.downunder` are checked at run time. For the second type, you can install them at any time (ensuring that python can find them) and they should work. For the first type, you need to modify the options file and recompile with scons. The rest of this section deals with this.

To avoid having to specify the options file each time you run scons, copy an existing `_options.py` file from the `scons/` or `scons/templates/` directories. Put the file in the `scons` directory and name it *yourmachinename*`_options.py`.[5]. For example: on a machine named toybox, the file would be `scons/toybox_options.py`.

Individual lines can be enabled/disabled, by removing or adding # (the python comment character) to the beginning of the line. For example, to enable OpenMP, change the line

```
#openmp = True
```

to

```
openmp = True
```

If you are using libraries which are not installed in the standard places (or have different names) you will need to change the relevant lines. A common need for this would be using a more recent version of the boost::python library. You can also change the compiler or the options passed to it by modifying the relevant lines.

MPI

If you wish to enable or disable MPI, or if you wish to use a different implementation of MPI, you can use the `mpi` configuration variable. You will also need to ensure that the `mpi_prefix` and `mpi_libs` variables are uncommented and set correctly. To disable MPI use, `mpi = 'none'`.

Python3

`esys.escript` works with `python3` but until recently, many distributions have not distributed python3 versions of their packages. You can try it out though by modifying or adding the following variables in your options file:

```
pythoncmd='python3'

usepython3=True

pythonlibname='whatever_your_python3_library_is_called'
```

[5] If the name has - or other non-alpha characters, they must be replaced with underscores in the filename

Testing

As indicated earlier, you can test your build using `scons py_tests`. Note however, that some features like `netCDF` are optional for using `esys.escript`, the tests will report a failure if they are missing.

2.4 Cleaning up

Once the build (and optional testing) is complete, you can remove everything except:

- bin
- esys
- lib
- doc
- CREDITS
- LICENSE
- README

The last three aren't strictly required for operation. The `doc` directory is not required either but does contain examples of escript scripts.

You can run escript using *path_to_escript_files*`/bin/run-escript`.
Where *path_to_escript_files* is replaced with the real path.

> *Optional step*
> You can add the escript `bin` directory to your `PATH` variable. The launcher will then take care of the rest of the environment.

2.5 Optional Extras

Some other packages which might be useful include:

- Lapack and UMFPACK — direct solvers (install the relevant libraries and enable them in the options file).
- support for the Silo file format (install the relevant libraries and enable them in the options file).
- VisIt — visualisation package. Can be used independently but our `weipa` library can make a Visit plug-in to allow direct visualisation of escript simulations.
- gmsh — meshing software used by our `pycad` library.
- Mayavi2 — another visualisation tool.

2.5.1 Trilinos

`esys.escript` now has some support for Trilinos[6] solvers and preconditioners. The most significant limitation is that the current Trilinos release does not support block matrices so `esys.escript` can only use Trilinos solvers for single PDEs (i.e. no PDE systems).

If your distribution does not provide Trilinos packages you can build a working version from source. (See Appendix B)

[6] https://trilinos.org/

APPENDIX
A

Required compiler features

Building escript from source requires that your c++ compiler supports at least the following features:

- `std::complex<>`
- Variables declared with type `auto`
- Variables declared with type `decltype(T)`
- `extern template class` to prevent instantiation of templates.
- `template class` *`classname<type>;`* to force instantiation of templates
- `isnan()` is defined in the `std::` namespace

The above is not exhaustive and only lists language features which are more recent that our previous baseline of c++99 (or which we have recently begun to rely on). Note that we test on up to date versions of `g++, icpc & clang++` so they should be fine.

Note that in future we may use c++14 features as well.

APPENDIX
B

Trilinos

In order to solve PDEs with complex coefficients, escript needs to be compiled with `Trilinos` support. This requires that your version of Trilinos has certain features enabled. Since some precompiled distributions of `Trilinos` are not built with these features, you may need to compile `Trilinos` yourself as well.

While we can't provide support for building `Trilinos`, we do list below, a configuration file which seems to work for under Debian 9 "stretch"[1].

B.1 Debian "stretch" configuration

B.1.1 Required packages

The following packages should be installed to attempt this build:

cmake

g++

libsuitesparse-dev

libmumps-dev

libboost-dev

libscotchparmetis-dev

libmetis-dev

libcppunit-dev

B.1.2 Configuration file

```
#!/bin/bash

# Set this to the root of your Trilinos source directory.
TRILINOS_PATH=../trilinos-12.12.1-Source

#
# You can invoke this shell script with additional command-line
# arguments.  They will be passed directly to CMake.
#
EXTRA_ARGS=$@
```

[1] At time of writing, `stretch` is still in testing, but is due to release this month

```
rm -f CMakeCache.txt

cmake \
  cmake -D MPI_C_COMPILER=`which mpicc` \
      -D MPI_CXX_COMPILER=`which mpic++` \
      -D MPI_Fortran_COMPILER=`which mpif77` \
      -D CMAKE_INSTALL_PREFIX=/usr/local/trilinos/ \
      -D Trilinos_ENABLE_CXX11=ON \
      -D Trilinos_ENABLE_Fortran=ON \
      -D BUILD_SHARED_LIBS=ON \
      -D TPL_ENABLE_BLAS=ON \
      -D TPL_ENABLE_LAPACK=ON \
      -D TPL_ENABLE_Boost=ON \
      -D TPL_ENABLE_Cholmod=ON \
      -D TPL_ENABLE_CppUnit=ON \
      -D TPL_ENABLE_Matio=ON \
      -D TPL_ENABLE_METIS=ON \
      -D TPL_ENABLE_MUMPS=ON \
      -D TPL_ENABLE_ParMETIS=ON \
      -D TPL_ENABLE_Pthread=ON \
      -D TPL_ENABLE_SCALAPACK=ON \
      -D TPL_ENABLE_UMFPACK=ON \
      -D TPL_BLAS_INCLUDE_DIRS=/usr/include/suitesparse \
      -D TPL_Cholmod_INCLUDE_DIRS=/usr/include/suitesparse \
      -D TPL_Cholmod_LIBRARIES='/usr/lib/x86_64-linux-gnu/libcholmod.so;/usr/lib/x86_64-
      -D TPL_UMFPACK_INCLUDE_DIRS=/usr/include/suitesparse \
      -D TPL_SCALAPACK_LIBRARIES=/usr/lib/libscalapack-openmpi.so \
      -D Trilinos_ENABLE_Amesos=ON \
      -D Trilinos_ENABLE_Amesos2=ON \
      -D Trilinos_ENABLE_Anasazi=OFF \
      -D Trilinos_ENABLE_AztecOO=ON \
      -D Trilinos_ENABLE_Belos=ON \
      -D Trilinos_ENABLE_Epetra=ON \
      -D Trilinos_ENABLE_EpetraExt=ON \
      -D Trilinos_ENABLE_Galeri=OFF \
      -D Trilinos_ENABLE_Ifpack=ON \
      -D Trilinos_ENABLE_Ifpack2=ON \
      -D Trilinos_ENABLE_Isorropia=OFF \
      -D Trilinos_ENABLE_Kokkos=ON \
      -D Trilinos_ENABLE_Komplex=ON \
      -D Trilinos_ENABLE_MueLu=ON \
      -D Trilinos_ENABLE_ML=OFF \
      -D Trilinos_ENABLE_Moertel=OFF \
      -D Trilinos_ENABLE_Teuchos=ON \
      -D Trilinos_ENABLE_Tpetra=ON \
      -D Trilinos_ENABLE_Thyra=OFF \
      -D Trilinos_ENABLE_Pliris=OFF \
      -D Trilinos_ENABLE_PyTrilinos=OFF \
      -D Trilinos_ENABLE_Zoltan=OFF \
      -D Trilinos_ENABLE_Zoltan2=OFF \
      -D Trilinos_ENABLE_EXPLICIT_INSTANTIATION=ON \
      -D Trilinos_ENABLE_COMPLEX=ON \
      -D Trilinos_ENABLE_OpenMP=ON \
      -D TPL_ENABLE_MPI=ON \
      -D Trilinos_DUMP_PACKAGE_DEPENDENCIES=ON \
```

```
      -D Trilinos_ENABLE_ALL_OPTIONAL_PACKAGES=ON \
      -D KOKKOS_ENABLE_AGGRESSIVE_VECTORIZATION=ON \
$EXTRA_ARGS \
$TRILINOS_PATH
```