Conversion de Modulef en Fortran 77

Patrick Laug

INRIA

CENTRE DE ROCQUENCOURT

Rapports Techniques

N° 34

CONVERSION DE MODULEF EN FORTRAN 77

Patrick LAUG

Avril 1984

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
BP 105
78153 Le Chesnay Cedex
France
Tél. (3) 954 90 20

Résumé

Ce rapport montre les difficultés rencontrées et les solutions adoptées pour convertir en Fortran 77 des programmes déjà écrits en Fortran 66. L'expérience porte sur la bibliothèque scientifique Modulef, qui comprend plus de 300 000 lignes de code.

Abstract

This report states the difficulties we met and the solutions we adopted to convert in Fortran 77 programs already written in Fortran 66. The experience concerns the Modulef scientific library, which contains more than 300 000 lines of code.

Février 1984

Patrick Laug

CONVERSION DE MODULEF EN FORTRAN 77

Sommaire

1. Introduction
2. Remplacement des constantes Hollerith par des chaînes de caractères
3. DATA avec caractères
4. CALL avec caractères
5. DIMENSION des paramètres
6. Améliorations diverses
7. Conclusion

Bibliographie

Annexe : listings d'utilitaires nouvellement définis

Notations

Dans les exemples, les instructions en Fortran 66 (resp. 77) sont repérées par les caractères .....66 (resp. .....77) en fin de ligne.

1. Introduction

Durant les années 70, les compilateurs se conformaient à la norme "Fortran 66". Depuis, l'ANSI a défini la norme "Fortran 77", qui est mieux spécifiée et plus puissante. Fortran 66 étant amené à disparaître, il était nécessaire que Modulef réagisse rapidement. Un groupe de travail s'est constitué, réunissant une trentaine de personnes en mars, juin, et octobre 83. Ces réunions ont permis de définir avec précision les modifications indispensables, ainsi que les améliorations possibles.

Actuellement, une version de Modulef en Fortran 77 est disponible. Le but de la présente brochure est de présenter les différences entre la version 3.4 de juin 1983 et cette version. Les modifications ont été apportées avec un double objectif :

- changer le moins possible les habitudes de programmation (compatibilité) : les mêmes modules sont fournis, avec toutefois certaines corrections mineures ("fiches d'incidents"). Les quelques changements dans la spécification des programmes sont détaillés dans cette brochure.

- respecter au maximum la norme Fortran 77 : en fait, le respect absolu de la norme ANSI aurait exigé des transformations trop importantes. Les principaux points de divergence sont les suivants :

  - La norme ANSI X3.9-1978 précise page 15-16 : "Association of Dummy and Actual Arguments. (...) A valid association occurs only if the type of the actual argument is the same as the type of the corresponding dummy argument". En pratique, nous avons dû accepter le mélange des types des paramètres numériques, afin de maintenir l'emploi d'un super-tableau unique contenant des variables de type INTEGER, REAL, DOUBLE PRECISION, ... Par contre, la norme est totalement respectée pour les paramètres de type CHARACTER.

  - On suppose qu'une variable de type INTEGER contient suffisamment de bits pour coder 4 caractères (cf fonctions CHAR4 et ICHAR4).

Malgré ces points de divergence, la version actuelle est exploitable sur la plupart des compilateurs Fortran 77, comme l'ont montré des tests sur Apollo, Bull, CDC, Cray, Data General, Harris, IBM, ICL, Norsk Data, Prime, SM 90, Univac, Vax, ... Seules les bibliothèques MEFGREL et MEFTARJ n'ont pas été modifiées, car de nouvelles versions seront fournies prochainement par leurs auteurs respectifs.

2. Remplacement des constantes Hollerith par des chaînes de caractères

En Fortran 77, les constantes Hollerith ne sont pas admises, car le type CHARACTER est plus général. Nous avons donc remplacé toutes les constantes de la forme :

```
      4HXXXX                                                    .....66
```

par :

```
      'XXXX'                                                    .....77
```

Mais, dans les programmes, ces constantes étaient associées à des variables de type INTEGER. Nous avons donc défini des utilitaires de conversion INTEGER - CHARACTER*4 (listings en annexe) :

CHAR4 est une fonction qui retourne une chaîne de 4 caractères à partir d'un paramètre entier.

ICHAR4 est la fonction inverse de CHAR4, qui retourne un entier à partir d'une chaîne de 4 caractères.

TRCHTA est un sous-programme tel que CALL TRCHTA(C,M,L) transfère la chaîne C dans le tableau entier M de L mots (4 caractères par mot).

La fonction NOMAFF, qui retournait un entier à partir d'une constante Hollerith, a été supprimée pour éviter un double emploi avec ICHAR4.

A l'aide des utilitaires CHAR4, ICHAR4, TRCHTA, nous avons modifié les instructions où des chaînes de caractères étaient utilisées : DATA (cf paragraphe 3) et CALL (cf paragraphe 4).

3. DATA avec caractères

L'instruction DATA permet d'initialiser des scalaires ou des tableaux, et nous distinguerons trois cas :

```
      DATA scalaire /constante  caractère/
      DATA tableau  /constantes caractères seulement/
      DATA tableau  /constantes entières et caractères/ .
```

3.1 DATA scalaire /constante caractère/

Les instructions de la forme :

```
      DATA I /'XXXX'/                                           .....66
```

ont été détruites, et on a inséré avant la première instruction exécutable :

```
      I = ICHAR4('XXXX')                                        .....77
```

3.2 DATA tableau /constantes caractères seulement/

Exemple :

```
      INTEGER M(3)                                                    .....66
      DATA M /'XXXX','YYYY','ZZZZ'/                                   .....66
```

a été remplacé par :

```
      CHARACTER*4 M(3)                                                .....77
      DATA M /'XXXX','YYYY','ZZZZ'/                                   .....77
```

et la cohérence des types a été rétablie à l'aide de CHAR4 et ICHAR4. Ces modifications ont été effectuées manuellement, en s'efforçant de sortir les invariants des boucles. Par exemple, dans :

```
      INTEGER MOT                                                     .....77
      CHARACTER*4 CHAR4,M(L)                                          .....77
      DO 100 I = 1,L                                                  .....77
         IF (CHAR4(MOT).EQ.M(I)) GOTO ...                             .....77
  100 CONTINUE                                                        .....77
```

CHAR4(MOT) est invariant dans la boucle 100 et, pour éviter des appels répétés, il est préférable d'écrire :

```
      INTEGER MOT                                                     .....77
      CHARACTER*4 CHAR4,M(L),CMOT                                     .....77
      CMOT = CHAR4(MOT)                                               .....77
      DO 100 I = 1,L                                                  .....77
         IF (CMOT.EQ.M(I)) GOTO ...                                   .....77
  100 CONTINUE                                                        .....77
```

3.3 DATA tableau /constantes entières et caractères/

Selon le cas, le tableau a été maintenu de type INTEGER ou converti en CHARACTER*4 :

3.3.1 Conservation du type INTEGER

Exemple (tiré de la subroutine ED2P1D) :

```
      DIMENSION MD(36)                                                .....66
      DATA MD /35,'ELAS','TRIA','2P1D',200001,3,3,3,0,2,2,1,1,0,1,    .....66
     é   'DEPL','ACEM','ENT ','EN X','DEPL','ACEM','ENT ','EN Y'      .....66
     é   2,1,1,2,1,1,1,1,1,1,1,1,1/                                   .....66
```

a été remplacé par :

```
      DIMENSION MD(36)                                                .....77
      DATA MD /35,0,0,0,200001,3,3,3,0,2,2,1,1,0,1,                   .....77
     é   0,0,0,0,0,0,0,0,                                             .....77
     é   2,1,1,2,1,1,1,1,1,1,1,1,1/                                   .....77
      MD(2) = ICHAR4('ELAS')                                          .....77
      MD(3) = ICHAR4('TRIA')                                          .....77
      MD(4) = ICHAR4('2P1D')                                          .....77
      CALL TRCHTA('DEPLACEMENT EN XDEPLACEMENT EN Y', MD(16), 8)      .....77
```

3.3.2 Conversion au type CHARACTER*4

Exemple (tiré de la subroutine DOBELA) :

```
      DIMENSION NB(101)                                          .....66
      DATA NB/100,                                               .....66
     é  'TRIA','2P1D','TRIA','2P2D','TRIA','2P2C',.../           .....66
```

a été remplacé par :

```
      PARAMETER (LNB=50)                                         .....77
      CHARACTER*8 NB(LNB)                                        .....77
      DATA NB /                                                  .....77
     é  'TRIA2P1D','TRIA2P2D','TRIA2P2C',.../                    .....77
```

et le sous-programme a été adapté manuellement au nouveau type du tableau.

A titre de remarque, il était également possible de remplacer l'entier 100 par la chaîne ' 100' et de décoder cette chaîne par un READ interne.

4. CALL avec caractères

Soit un sous-programme (s.p.), dit s.p. appelant, contenant une instruction de la forme :

```
      CALL SP('XXXX')                                            .....66
```

Supposons maintenant que le s.p. appelé soit spécifié par :

```
      SUBROUTINE SP(I)                                           .....66
```

Si l'on maintenait ces instructions en Fortran 77, le paramètre effectif serait de type CHARACTER, alors que le paramètre formel serait de type INTEGER. Cela n'est pas admis par la norme ANSI X3.9-1978 (page 15-16), et risque effectivement de provoquer des erreurs d'exécution, comme l'ont montré des tests sur différents types de machines. Deux possibilités permettent de remédier à ce problème :

a) Ne pas modifier le s.p. appelé, et remplacer :

```
      CALL SP('XXXX')                                            .....66
```

par :

```
      CALL SP(ICHAR4('XXXX'))                                    .....77
```

b) Modifier le s.p. appelé, en remplaçant :

```
      SUBROUTINE SP(I)                                           .....66
```

par :

```
      SUBROUTINE SP(C)                                           .....77
      CHARACTER*4 C                                              .....77
      I = ICHAR4(C)                                              .....77
```

Cette solution a l'avantage de ne pas modifier les instructions où SP est appelé avec une chaîne de caractères (légèreté d'écriture), et autorise facilement des extensions ultérieures pour un nombre de caractères différents de 4. Par contre, s'il existe aussi des appels de SP avec un entier, ceux-ci doivent être modifiés, et l'écriture est allourdie :

```
      CALL SP(I)                                                 .....66
```

devient alors :

```
      CHARACTER*4 CHAR4                                          .....77
      CALL SP(CHAR4(I))                                          .....77
```

Il a donc fallu choisir entre les solutions (a) et (b), selon l'usage des s.p. concernés. Finalement, la solution (b) a été retenue uniquement pour les s.p. ci-dessous. Il s'agit principalement des s.p. de gestion des tableaux dynamiques et des S.D. Modulef.

Les paramètres qui sont maintenant de type CHARACTER sont soulignés. On écrira par exemple :

```
      CALL ADRESS(1,'XXXX',IAXXXX,LXXXX,M)                    .....77
ou :
      CHARACTER*4 CHAR4                                       .....77
      INTEGER NXXXX                                           .....77
      CALL ADRESS(1,CHAR4(NXXXX),IAXXXX,LXXXX,M)              .....77
```

Tableau des s.p. ayant maintenant des paramètres de type CHARACTER :

```
ADRESS : SUBROUTINE ADRESS(NTYPE,NOMTAB,IA,L,M)
CARACD : INTEGER FUNCTION CARACD(CARAC)
CARACG : INTEGER FUNCTION CARACG(CARAC)
COPISD : SUBROUTINE COPISD(M,NOMSD,NOMTAB,NIV1,NC1,IAC1,NIV2,NC2,IAC2)
DILATE : SUBROUTINE DILATE(NOMTAB,IA,L,M)
DLBLOQ : SUBROUTINE DLBLOQ(M,NFNDL1,ND,NOE,NFBDCL,NIBDCL,
         é   NNOBL,IANOBL,LNOBL,NVABL,IAVABL,LVABL)
IEGAL  : FUNCTION IEGAL(K,I1,I2)
IMPRIA : SUBROUTINE IMPRIA(NOMTAB,NTY,K1,K2,IA,AR,AD,AC)
IMPTAB : SUBROUTINE IMPTAB(M,ML,XM,DM,NTYP,IA,L,NOMTAB)
INCANO : FUNCTION INCANO(NOM,NB,ICAR)
INCAPA : FUNCTION INCAPA(NOM,KAR,ICAR)
INICSD : SUBROUTINE INICSD(M,NOMSD,NIVSD,NCOMMO,LCOMMO,NCSD,IACSD)
INITMP : SUBROUTINE INITMP(M,NUMPAG,NBMAPA,LPAGE,NTY,N1P,IA1P,IMPRE,
         é   N,IANPAG)
INTABO : SUBROUTINE INTABO(NOMSD,NIVSD,NTBASD,NTABO)
NIVLIB : SUBROUTINE NIVLIB(NOMSD,NILIB,NOMBRE)
NOM    : SUBROUTINE NOM(N,N1)
NOMLON : SUBROUTINE NOMLON(N,N1,L,L1)
READRE : SUBROUTINE READRE(NTYPE,NOMTAB,IA,L,M,NSORTI)
RENOMM : SUBROUTINE RENOMM(NTAB1,NTAB2,IATAB2,LTAB2,M)
RESOUR : SUBROUTINE RESOUR(NOPT,NOMOD1,NOMOD2,NOMSD,NTSD,MCMSD,MCOSD,
         é   NFSD,NESD,NMNFSD)
RSTSDE : SUBROUTINE RSTSDE(M,NOMSDE,NFSDE,NIVSDE,NCOMMO,LCOMMO,
         é   LESDE,RESDE,NCSDE,IACSDE)
SAUCSD : SUBROUTINE SAUCSD(M,NOMSD,NIVSD,NCOMMO,LCOMMO,NCSD,IACSD)
SAUSDS : SUBROUTINE SAUSDS(M,NOMSDS,NFSDS,NIVSDS,NCOMMO,LCOMMO,
         é   NCSDS,IACSDS,ECSDS,NOPFI,NMOSDS)
TROUVE : SUBROUTINE TROUVE(NTYPE,NOMTAB,IA,L,NUMC,M)
TUER   : SUBROUTINE TUER(NOMTAB,M)
TUERSD : SUBROUTINE TUERSD(M,NOMSD,NIVSD,NOPT)
VERIF  : SUBROUTINE VERIF(M,IA,NFSD,NSD)
```

Les appels aux s.p. graphiques ont provisoirement été laissés tels quels. Par exemple, pour le s.p. Benson PCARA, faut-il passer en paramètre une chaîne de caractères ou un tableau d'entiers ? La documentation date encore de Fortran 66...

5. DIMENSION des paramètres

En Fortran 66, la dernière borne de tout paramètre formel était ignorée, et il était fréquent d'y donner une valeur quelconque, en général 1 :

```
      SUBROUTINE SP(X,Y)                                          .....66
      DIMENSION X(1),Y(6,1)                                       .....66
      R = X(3) + Y(5,2)                                           .....66
```

En Fortran 77, la plupart des compilateurs rejettent cette notation. En effet, il est possible de déclarer un paramètre de taille inconnue ("assumed-size dummy array") par l'instruction :

```
      DIMENSION X(*),Y(6,*)                                       .....77
```

Pour convertir la version Modulef 3.4, il aurait été possible d'utiliser cette notation pour toutes les déclarations de tableaux formels. Cependant, si la taille effective était connue (expression contenant des constantes, des paramètres, ou des communs : cf "constant or adjustable array declarator", norme X3.9-1978 page 5-2), on s'est efforcé de la conserver, ce qui présente plusieurs avantages :

- meilleure lisibilité,
- vérification des indices à la compilation ou à l'exécution,
- optimisation sur les compilateurs vectoriels,
- entrées/sorties plus faciles à programmer (ordres Fortran READ et WRITE),
- impressions simplifiées dans les programmes ou avec les "debuggers" symboliques.

6. Améliorations diverses

L'arrivée de Fortran 77 a exigé de nombreuses mises à jour, mais les nouvelles possibilités de ce langage ont permis de réécrire des sous-programmes autrefois non portables ou trop complexes : fonctions INCAPA et INCANO, utilitaires d'accès direct, format libre.

Par ailleurs, les réunions Fortran 77 ont permis de définir plusieurs améliorations : fonctions INFO, allocation dynamique, réorganisation des bibliothèques, structuration des bandes, insertion d'un nouvel élément fini, suppression des RETURN finals.

6.1 Fonctions INCAPA et INCANO

Grâce au traitement des chaînes de caractères en Fortran 77, la fonction INCAPA(NOM,CAR,ICAR) a été réécrite de manière portable (listing en annexe). Rappelons que les deux premiers paramètres sont maintenant de type CHARACTER (cf tableau paragraphe 4).

En fait, on peut maintenant éviter d'utiliser INCAPA et INCANO :

```
      CHARACTER NOMSD*4,CHAR4*4                                   .....77
      NC = INCANO(NOMSD,NIVEAU,3)                                 .....77
      NC = INCAPA(CHAR4(NC),'%',4)                                .....77
```

s'écrit plus simplement :

```
      CHARACTER NOMSD*4,NUMALP*1                                  .....77
      NC = ICHAR4(NOMSD(1:2) // NUMALP(NIVEAU) // '%')            .....77
```

6.2 Accès direct

A ce jour, les utilitaires de gestion de fichiers en accès direct respectent totalement la norme Fortran 77. Quelques modifications sont nécessaires sur les sites qui ne respectent pas totalement cette norme. Par exemple, dans DEFDIR, il faut ajouter à l'ordre OPEN le mot-clé MAXREC sur Vax.

L'utilisateur peut maintenant manipuler des fichiers en accès direct de deux façons :

- comme autrefois, c'est-à-dire en appelant les s.p. DEFDIR, ECRDIR, LECDIR, OUVDIR, ... (cf brochure Modulef no 42). Une restriction importante a été nécessaire : SI ON UTILISE LES UTILITAIRES D'ACCES DIRECT, LE SUPER-TABLEAU DOIT OBLIGATOIREMENT SE TROUVER DANS LE COMMUN BLANC.

- en utilisant directement les ordres Fortran READ, WRITE, ... (s'inspirer éventuellement des utilitaires d'accès direct). Les numéros de page sont compatibles, c'est-à-dire que :

```
      CALL LEPDIR(IU,IP,T,L)                                    .....77
est équivalent à :
      READ (UNIT=IU,REC=IP) (T(I),I=1,L)                        .....77
```

Les assignations des fichiers en accès direct peuvent se faire :

- par le langage de commande de la machine (JCL). Par exemple, sur Multics, on écrit :

```
      att 010 toto
```

ce qui équivaut à :

```
      io attach file10 vfile_ toto -blocked 1024
```

- par un OPEN Fortran :

```
      OPEN (UNIT=IU,FILE='TOTO',ACCESS='DIRECT',FORM='UNFORMATTED')  .....77
```

Cette dernière possibilité est tout particulièrement utile pour l'écriture de programmes conversationnels.

6.3 Format libre

Fortran 77 permet de décoder de manière simple des valeurs numériques, grâce aux "fichiers internes" (norme ANSI X3.9-1978, page 12-5). L'ancienne version du format libre a ainsi été considérablement allégée.

Le programmeur a maintenant la possibilité d'utiliser des s.p. différents selon le type de la variable à lire, au lieu du s.p. unique LECTU1. On peut ainsi respecter le contrôle des types effectué par certains compilateurs, et lire des variables de type CHARACTER :

```
CALL LIBENT(V) équivaut à CALL LECTU1(1,V)    (entier)
CALL LIBRSP(V)     "      CALL LECTU1(2,V)    (réel simple précision)
CALL LIBLOG(V)     "      CALL LECTU1(3,V)    (logique)
CALL LIBHOL(V)     "      CALL LECTU1(4,V)    (Hollerith)
CALL LIBRDP(V)     "      CALL LECTU1(5,V)    (réel double précision)
CALL LIBCSP(V)     "      CALL LECTU1(6,V)    (complexe simple précision)
CALL LIBCDP(V)     "      CALL LECTU1(7,V)    (complexe double précision)
CALL LIBCAR(V) lit la donnée suivante dans une variable CHARACTER
CALL LIBLIG(V) lit la ligne suivante dans une variable CHARACTER
```

De plus, les améliorations suivantes ont été apportées aux utilisateurs :

- La syntaxe pour les réels est plus souple. En particulier, on peut utiliser la notation entière (par exemple, 36) au lieu de la notation réelle avec un point décimal (36.).

- Avant chaque donnée devant être lue, on peut taper une ou plusieurs des commandes suivantes (n désigne un entier positif ou nul) :

%?		: imprime le type de la donnée à lire
%ADRESSAGE	n	: modifie le taux d'impression des adressages (NNN = n)
%BAVARDAGE	n	: modifie le taux d'impression des modules (IMPRE = n)
%ECHO	n	: modifie le taux d'impression des lignes lues (IMPLEC = n)
%IMPRIMANTE	n	: modifie le numéro logique de l'imprimante (IMPRIM = n)
%LECTEUR	n	: modifie le numéro logique du lecteur (LECTEU = n) Les lectures suivantes s'effectuent sur le fichier n, en partant de la position courante sur ce fichier, puis reviennent au lecteur précédent en fin de fichier ou sur une ligne %RETOUR (voir aussi %OUVRIR).
%OUVRIR	n	: positionne le fichier n avant le premier enregistrement (REWIND), puis effectue la même action que %LECTEUR.
%RETOUR		: retourne au fichier précédent. S'il n'y en a pas, arret de l'exécution du programme (STOP).
%SAUVER	n	: recopie sur le fichier n tout ce qui est tapé sur le lecteur initial.
%SOUFFLEUR	n	: si n = 0, pas de souffleur : si n = 1, impression du type de la donnée à taper avant chaque lecture de ligne.
%TERMINAL	n	: modifie le taux d'interactivité (INTERA = n).

Il suffit en fait de taper les premières lettres : %?, %A, %B, %E, %I, %L, %R, %SA, %SO, %T.

Les variables précédentes sont initialisées par le s.p. INITI.

- Toutes ces commandes peuvent être activées par le programme d'appel et non par les données, à l'aide de l'utilitaire LIBCMD. Par exemple,

```
      CALL LIBCMD('BAVARDAGE 4')                          .....77
```

a le même effet que si l'on avait dans les données :

```
      %BAVARDAGE 4                                        (donnée)
```

Ceci permet de modifier les paramètres usuels de manière simple pour l'utilisateur, en évitant l'emploi de communs, ce qui était autrefois obligatoire :

```
      COMMON /TRAVA1/ PAD1(29),IMPRE,PAD2(3)              .....66
      IMPRE = 4                                           .....66
```

6.4 Fonctions INFO

Ces fonctions permettent aux programmes d'accéder à des informations générales (numéro de l'imprimante, plus petit nombre flottant positif, nom de l'utilisateur, ...) sans avoir recours à des communs.

Par exemple, le programme :

```
      COMMON /UNITES/ LECTEU,IMPRIM,FILUNI(30)
      WRITE (IMPRIM,*) T                                    .....77
                                                            .....77
```

peut maintenant s'écrire :

```
      WRITE (IINFO('I'),*) T
                                                            .....77
```

Ces fonctions doivent être adaptées à chaque site (voir listings en annexe). Elles permettent actuellement de retourner des informations de type INTEGER, REAL, DOUBLE PRECISION, ou CHARACTER. Parmi ces informations, on peut citer (le mot-clé associé figure en partie gauche) :

- INTEGER FUNCTION IINFO(MOTCLE)

L ou LECTEUR	: numéro logique du lecteur	(LECTEU)
I ou IMPRIMANTE	: numéro logique de l'imprimante	(IMPRIM)
INTERACTIVITE	: 0 batch, 1 interactif, 2 graphique	(INTERA)
BAVARD	: niveau d'impression	(IMPRE)

- REAL FUNCTION RINFO(MOTCLE)

PETIT	: plus petit réel positif en simple précision
GRAND	: plus grand réel en simple précision

- DOUBLE PRECISION FUNCTION DINFO(MOTCLE)

PETIT	: plus petit réel positif en double précision
GRAND	: plus grand réel en double précision
HORLOGE	: temps horloge à partir d'un instant fixé
DELTA HORLOGE	: accroissement du temps horloge
CPU	: temps cpu à partir d'un instant fixé
DELTA CPU	: accroissement du temps cpu

- CHARACTER*80 FUNCTION KINFO(MOTCLE)

MACHINE	: nom de la machine hôte
TITRE	: titre du travail (demandé par INITI)
DATE	: date du jour (aammjj)
HEURE	: heure (hhmmssffffff)
UTILISATEUR	: nom de l'utilisateur

Ces fonctions seront étendues selon les besoins.

6.5 Allocation dynamique

Pour permettre des extensions futures, le super-tableau M est ajouté à la liste des paramètres de certains utilitaires d'allocation dynamique :

```
SUBROUTINE IMATAB (M)
SUBROUTINE TUER   (NOMTAB,M)
SUBROUTINE TROUVE (NTYPE,NOMTAB,IA,L,NUMC,M)
SUBROUTINE RENOMM (NTAB1,NTAB2,IATAB2,LTAB2,M)
```

En conséquence, le paramètre M est aussi ajouté au sous-programme de fonctions interprétées :

```
SUBROUTINE FONTER(M)
```

6.6 Réorganisation des bibliothèques

Les utilitaires devenus portables ont été mis dans MEFUTIL.

Par ailleurs, la bibliothèque MEFELAS étant jugée trop volumineuse, celle-ci a été découpée en :

MEFELAU :	utilitaires de base
MEFELA2 :	élasticité 2D
MEFELA3 :	élasticité 3D
MEFELAG :	sous-programmes généraux appelant MEFELA2 et MEFELA3

6.7 Structuration des bandes

Le groupe de travail Fortran 77 a cherché un moyen de faciliter l'échange des bandes magnétiques Modulef. Comme à l'accoutumée, la structure des bandes est compatible avec le processeur IEBUPDTE de IBM. Ce processeur ne reconnaît que les mots-clés ADD, REPL, et CHANGE, mais autorise une zone de commentaires (après un ou plusieurs blancs).

Chaque bibliothèque commence par une ligne de la forme :
./BIBLIxxx ADD NAME=nom,SSI=aammjjhh commentaire

Chaque membre commence par une ligne de la forme :
C/MEMBRxxx ADD NAME=nom,SSI=aammjjhh commentaire

xxx décrit le type de la bibliothèque ou du membre :

F66 Fortran 66	DOC documentation
F77 Fortran 77	JCL langage de commande batch
PL1 PL/1	TSO langage de commande interactif
ASM assembleur	DIV divers
DAT données	

nom nom de la bibliothèque ou du membre

aammjjhh date et heure de la version

commentaire (non implémenté) contiendra l'un des mots-clés suivant :

- ADD : bibliothèque ou membre à ajouter sur le site,
- REP : bibliothèque ou membre à remplacer,
- DEL : bibliothèque ou membre à détruire,
- OLD : bibliothèque ou membre inchangé depuis la dernière version.

6.8 Insertion d'un nouvel élément fini

La manière d'insérer un nouvel élément dans Modulef a été simplifiée :

- Les s.p. DONTCN et DONTNC sont supprimés, et les s.p. DOBIEL, DOTABD, DOBELA, DOBTHE, DOBFLU, DOBELN sont modifiés afin de remplacer DONTCN et DONTNC.

- Lorsqu'on veut insérer un nouvel élément dans une bibliothèque "BIBL" existante, outre l'appel au sous-programme de définition à mettre dans le sous-programme DOB"BIBL", il faut ajouter le nom de l'élément dans le DATA NB et augmenter de 1 la longueur LNB (qui figure dans une instruction PARAMETER, cf paragraphe 3.3.2).

- Pour introduire une nouvelle bibliothèque "BIBN", il faut introduire le nom dans le DATA NA du s.p. DOTABD, créer le s.p. DOB"BIBN", et modifier DOBIEL.

Pour tout renseignement, s'adresser à D. Steer, INRIA.

6.9 Suppression des RETURN finals

Pour diminuer le nombre total d'instructions, les séries d'instructions :

```
      RETURN                                          .....77
      END                                             .....77
```

ont été remplacées par :

```
      END                                             .....77
```

7. Conclusion

La version Modulef 3.4 - Fortran 77 est maintenant opérationnelle. Cependant, quelques problèmes subsistent : comment passer un texte en paramètre d'un sous-programme graphique (cf paragraphe 4) ? L'emploi du commun blanc est-il judicieux ? Sur les machines où les variables de type REAL occupent déjà 64 bits (CDC), comment exploiter les sous-programmes écrits en DOUBLE PRECISION ? Les performances de la gestion des fichiers en accès direct sont-elles trop dégradées ? Comment hiérarchiser les bibliothèques ? Des implémentations sur des sites divers soulèveront probablement d'autres questions. Une nouvelle réunion du groupe Fortran 77 permettrait d'exposer et de résoudre au mieux ces problèmes.

Le programmeur devra veiller aux quelques incompatibilités avec l'ancienne version, notamment le type des paramètres de certains sous-programmes (tableau paragraphe 4), et le super-tableau à mettre dans le commun blanc pour les utilitaires d'accès direct.

En revanche, il est maintenant possible de profiter des extensions Modulef (commandes en format libre, fonctions INFO, ...) et d'utiliser toutes les facilités de Fortran 77 (voir les exemples précédents, les documents cités dans la bibliographie, et l'annexe). Notons tout particulièrement :
- le type CHARACTER (utilisation de la fonction intrinsèque INDEX pour la recherche de mots-clés),
- les tableaux à bornes quelconques : DIMENSION A(-8:+8), B(2*M+N)
- les structures de contrôle :
  - IF THEN ELSE permet d'écrire des programmes plus lisibles, en évitant les "astuces" de programmation et la présence excessive d'étiquettes,
  - DO avec pas négatif,
- les entrées/sorties : format *, fichiers internes, fichiers en accès direct, OPEN, INQUIRE,
- la définition de constantes avec l'instruction PARAMETER.

Bibliograhie

- Ancienne norme (Fortran 66) : ANSI X3.9 - 1966 "Fortran".

- Nouvelle norme (Fortran 77) : ANSI X3.9 - 1978 "Programming Language Fortran" ("full language" sur les pages de droite).

- Comparaison entre ces deux normes :
  - W. Brainerd, "Fortran 77",
    Communication of the ACM, vol 21, no 10, octobre 1978.
  - B. Meyer, "La nouvelle norme Fortran 77",
    Atelier Logiciel no 10 (EDF Clamart), juin 1980.

- Compte rendus du groupe Fortran 77 (24 mars, 10 juin, et 13 octobre 1983).

CHAR4 (portable)

```
      CHARACTER*4 FUNCTION CHAR4(ICHAR4)
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C BUT :
C     REALISER LA CONVERSION : INTEGER A4 -> CHARACTER*4
C
C PARAMETRE D'ENTREE :
C     ICHAR4  : ENTIER CONTENANT 4 CARACTERES
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C PROGRAMMEUR : PATRICK LAUG : INRIA (3) 954 90 20 POSTE 3508
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      INTEGER ICHAR4
      CHARACTER*4 BUFFER
      WRITE (BUFFER,'(A4)') ICHAR4
      CHAR4 = BUFFER
      END
```

DINFO (non portable)

```
      DOUBLE PRECISION FUNCTION DINFO(MOTCLE)
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C BUT :
C     RETOURNER UNE INFORMATION DE TYPE DOUBLE PRECISION
C     *** VERSION MULTICS ***
C
C PARAMETRE D'ENTREE :
C     MOTCLE  : NATURE DE L'INFORMATION A RETOURNER, EN CLAIR
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C PROGRAMMEUR : PATRICK LAUG : INRIA (3) 954 90 20 POSTE 3508
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      CHARACTER*(*) MOTCLE
      DATA PETIT /040040000000000000000000/
      DATA GRAND /037677777777777777777777/
C
C PLUS PETIT REEL POSITIF
      IF (MOTCLE .EQ. 'PETIT') THEN
         DINFO = PETIT
C
C PLUS GRAND REEL
      ELSE IF (MOTCLE .EQ. 'GRAND') THEN
         DINFO = GRAND
C
C TEMPS HORLOGE A PARTIR D'UN INSTANT FIXE
C SUR MULTICS, LA VALEUR RETOURNEE EST LE NOMBRE DE MICROSECONDES
C ECOULEES DEPUIS LE 1ER JANVIER 1901
      ELSE IF (MOTCLE .EQ. 'HORLOGE') THEN
         DINFO = DINFO_$HORLOGE( )
C
C ACCROISSEMENT DU TEMPS HORLOGE
C - AU PREMIER APPEL, RETOURNE LA MEME VALEUR QUE DINFO('HORLOGE')
C - ENSUITE, RETOURNE LA DIFFERENCE ENTRE LA VALEUR COURANTE DE
C   L'HORLOGE ET LA VALEUR QU'ELLE AVAIT A L'APPEL PRECEDENT
C REMARQUE : POUR QUE LE RESULTAT AIT UN SENS, L'UTILISATEUR DOIT
C BIEN LOCALISER TOUS LES APPELS A DINFO('DELTA HORLOGE')
      ELSE IF (MOTCLE .EQ. 'DELTA HORLOGE') THEN
         DINFO = DINFO_$DELTA_HORLOGE( )
```

```
C
C TEMPS CPU A PARTIR D'UN INSTANT FIXE
C SUR MULTICS, LA VALEUR RETOURNEE EST LE TEMPS CPU (EN MICROSECONDES)
C UTILISE PAR LE PROCESSUS APPELANT, DEPUIS QU'IL A ETE CREE
      ELSE IF (MOTCLE .EQ. 'CPU') THEN
         DINFO = DINFO_$CPU()
C
C ACCROISSEMENT DU TEMPS CPU
C - AU PREMIER APPEL, RETOURNE LA MEME VALEUR QUE DINFO('CPU')
C - ENSUITE, RETOURNE LA DIFFERENCE ENTRE LA VALEUR COURANTE DU
C   TEMPS CPU ET LA VALEUR QU'IL AVAIT A L'APPEL PRECEDENT
C REMARQUE : POUR QUE LE RESULTAT AIT UN SENS, L'UTILISATEUR DOIT
C BIEN LOCALISER TOUS LES APPELS A DINFO('DELTA CPU')
      ELSE IF (MOTCLE .EQ. 'DELTA CPU') THEN
         DINFO = DINFO_$DELTA_CPU()
C
C ERREUR
      ELSE
         WRITE (IINFO('I'),*) 'DINFO : MOT-CLE INCONNU : ', MOTCLE
         DINFO = 0.
      END IF
      END
```

ICHAR4 (portable)

```
      INTEGER FUNCTION ICHAR4(CHAR4)
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C BUT :
C     REALISER LA CONVERSION : CHARACTER*4 -> INTEGER A4
C
C PARAMETRE D'ENTREE :
C     CHAR4    : CHAINE DE 4 CARACTERES A CONVERTIR
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C PROGRAMMEUR : PATRICK LAUG : INRIA (3) 954 90 20 POSTE 3508
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      CHARACTER*(*) CHAR4
      CHARACTER*4 BUFFER
      BUFFER = CHAR4
      READ (BUFFER,'(A4)') I
      ICHAR4 = I
      END
```

IINFO (non portable)

```
      INTEGER FUNCTION IINFO(MOTCLE)
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C BUT :
C     RETOURNER UNE INFORMATION DE TYPE INTEGER
C     *** VERSION MULTICS ***
C
C PARAMETRE D'ENTREE :
C     MOTCLE  : NATURE DE L'INFORMATION A RETOURNER, EN CLAIR
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C PROGRAMMEUR : PATRICK LAUG - INRIA (3) 954 90 20 POSTE 3508
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
      CHARACTER*(*) MOTCLE
      COMMON /TRAVA1/ PADTR1(29),IMPRE,PADTR2(3)
      COMMON /UNITES/ LECTEU,IMPRIM,INTERA,NFNEWS,NFSAUV,PADUNI(27)
C
C 1. PARTIE PORTABLE
C ------------------
C
C NUMERO LOGIQUE DE L'IMPRIMANTE
C CE TEST EST PLACE EN TETE CAR IL EST LE PLUS FREQUEMMENT VRAI
      IF (MOTCLE .EQ. 'I' .OR. MOTCLE .EQ. 'IMPRIMANTE') THEN
         IINFO = IMPRIM
C
C NUMERO LOGIQUE DU LECTEUR
      ELSE IF (MOTCLE .EQ. 'L' .OR. MOTCLE .EQ. 'LECTEUR') THEN
         IINFO = LECTEU
C
C INTERACTIVITE
      ELSE IF (MOTCLE .EQ. 'INTERACTIVITE') THEN
         IINFO = INTERA
C
C NOUVELLES
      ELSE IF (MOTCLE .EQ. 'NOUVELLES') THEN
         IINFO = NFNEWS
C
C SAUVEGARDE
      ELSE IF (MOTCLE .EQ. 'SAUVEGARDE') THEN
         IINFO = NFSAUV
C
C BAVARDAGE
      ELSE IF (MOTCLE .EQ. 'BAVARD') THEN
         IINFO = IMPRE
C
C 2. PARTIE NON PORTABLE
C ----------------------
C
C NUMERO LOGIQUE DU LECTEUR INITIAL
      ELSE IF (MOTCLE .EQ. 'LECTEUR INITIAL') THEN
         IINFO = 5
C
C NUMERO LOGIQUE DE L'IMPRIMANTE INITIALE
      ELSE IF (MOTCLE .EQ. 'IMPRIMANTE INITIALE') THEN
         IINFO = 6
C
C DANS UN ORDRE OPEN POUR DES ENTREES/SORTIES NON FORMATTEES,
C LE MOT-CLE RECL DEFINIT UNE LONGUEUR MESUREE EN UNE UNITE
C QUI DEPEND DU PROCESSEUR (NORME FORTRAN 77, P. 12-20, L. 1 A 16).
C ON RETOURNE ICI LA TAILLE D'UN MOT-MACHINE EXPRIMEE DANS CETTE UNITE.
C SUR MULTICS, L'UNITE EST LE CARACTERE ET 1 MOT = 4 CARACTERES
      ELSE IF (MOTCLE .EQ. 'UNITE RECL') THEN
         IINFO = 4
C
C INTERACTIVITE INITIALE : 0 BATCH,
C                          1 INTERACTIF ALPHANUMERIQUE,
C                          2 INTERACTIF GRAPHIQUE.
      ELSE IF (MOTCLE .EQ. 'INTERACTIVITE INITIALE') THEN
         IINFO = IINFO_$INTERACTIVITE_INITIALE()
C
C ERREUR
      ELSE
         WRITE (IMPRIM,*) 'IINFO : MOT-CLE INCONNU : ', MOTCLE
         IINFO = 0
      END IF
      END
```

INCAPA (portable)

```
      FUNCTION INCAPA(NOM,CAR,ICAR)
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C BUT :
C     REMPLACER LE ICAR-IEME CARACTERE DE NOM PAR CAR
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C PROGRAMMEUR : PATRICK LAUG : INRIA (3) 954 90 20 POSTE 3508
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      CHARACTER NOM*4, CAR*1, BUFFER*4
      BUFFER = NOM
      BUFFER(ICAR:ICAR) = CAR
      READ (BUFFER,'(A4)') I
      INCAPA = I
      END
```

KINFO (non portable)

```
      CHARACTER*80 FUNCTION KINFO(MOTCLE)
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C BUT :
C     RETOURNER UNE INFORMATION DE TYPE CHARACTER
C     *** VERSION MULTICS ***
C
C PARAMETRE D'ENTREE :
C     MOTCLE  : NATURE DE L'INFORMATION A RETOURNER, EN CLAIR
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C PROGRAMMEUR : PATRICK LAUG : INRIA (3) 954 90 20 POSTE 3508
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      CHARACTER*(*) MOTCLE
      CHARACTER*4   CHAR4
      CHARACTER*80  BUFFER,
        KINFO_$DATE_JOUR,KINFO_$HEURE,KINFO_$UTILISATEUR
      COMMON /TRAVA1/ MTITRE(20),PADTRA(13)
C
C 1. PARTIE PORTABLE
C ------------------
C
C TITRE
      IF (MOTCLE .EQ. 'TITRE') THEN
        WRITE (BUFFER,'(20A4)') MTITRE
        KINFO = BUFFER
C
C 2. PARTIE NON PORTABLE
C ----------------------
C
C NOM DE LA MACHINE HOTE
      ELSE IF (MOTCLE .EQ. 'MACHINE') THEN
        KINFO = 'BULL DPS 68 (MULTICS)'
C
C DATE DU JOUR (AAMMJJ)
C AA ANNEE, MM MOIS, JJ JOUR
      ELSE IF (MOTCLE .EQ. 'DATE') THEN
        KINFO = KINFO_$DATE_JOUR()
C
C HEURE (HHMMSSFFFFFF)
C HH HEURE DE 00 A 23, MM MINUTE DE 00 A 59, SS SECONDE DE 00 A 59,
C FFFFFF MICROSECONDE, DE 000000 A 999999
      ELSE IF (MOTCLE .EQ. 'HEURE') THEN
        KINFO = KINFO_$HEURE()
```

```
C
C NOM DE L'UTILISATEUR
      ELSE IF (MOTCLE .EQ. 'UTILISATEUR') THEN
         KINFO = KINFO_$UTILISATEUR( )
C
C ERREUR
      ELSE
         WRITE (IINFO('I'),*) 'KINFO : MOT-CLE INCONNU : ', MOTCLE
         KINFO = ' '
      END IF
      END
```

RINFO (non portable)

```
      REAL FUNCTION RINFO(MOTCLE)
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C BUT :
C     RETOURNER UNE INFORMATION DE TYPE REAL
C     *** VERSION MULTICS ***
C
C PARAMETRE D'ENTREE :
C     MOTCLE  : NATURE DE L'INFORMATION A RETOURNER, EN CLAIR
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C PROGRAMMEUR : PATRICK LAUG : INRIA (3) 954 90 20 POSTE 3508
C.............................................................................
      CHARACTER*(*) MOTCLE
      DATA PETIT /0400400000000/
      DATA GRAND /0376777777777/
C
C PLUS PETIT REEL POSITIF
      IF (MOTCLE .EQ. 'PETIT') THEN
         RINFO = PETIT
C
C PLUS GRAND REEL
      ELSE IF (MOTCLE .EQ. 'GRAND') THEN
         RINFO = GRAND
C
C ERREUR
      ELSE
         WRITE (IINFO('I'),*) 'RINFO : MOT-CLE INCONNU : ', MOTCLE
         RINFO = 0.
      END IF
      END
```

TRCHTA (portable) *

```
      SUBROUTINE TRCHTA(CHAINE,MTAB,LTAB)
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C BUT :
C     TRANSFERT D'UNE CHAINE DE CARACTERES DANS UN TABLEAU D'ENTIERS
C     SI LA CHAINE EST TROP COURTE, ELLE EST COMPLETEE PAR DES BLANCS
C     SI ELLE EST TROP LONGUE, ELLE EST TRONQUEE
C
C PARAMETRES D'ENTREE :
C     CHAINE  : CHAINE DE CARACTERES
C     LTAB    : NB DE MOTS DU TABLEAU MTAB (ATTENTION : 3E PARAMETRE)
C
C PARAMETRE DE SORTIE :
C     MTAB    : TABLEAU D'ENTIERS
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C PROGRAMMEUR : PATRICK LAUG : INRIA (3) 954 90 20 POSTE 3508
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      CHARACTER*(*) CHAINE
      INTEGER MTAB(LTAB)
C
      DO 100 ITAB = 1,LTAB
      ITAB4 = ITAB * 4
      IF (ITAB4 .LE. LEN(CHAINE)) THEN
         MTAB(ITAB) = ICHAR4(CHAINE(ITAB4-3 : ITAB4))
      ELSE IF (ITAB4-3 .LE. LEN(CHAINE)) THEN
         MTAB(ITAB) = ICHAR4(CHAINE(ITAB4-3 : LEN(CHAINE)))
      ELSE
         MTAB(ITAB) = ICHAR4('    ')
      END IF
  100 CONTINUE
      END
```

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique