
Programmes de tests de (fichiers) FA et LFI

----

1) Introduction

Dans le cadre du projet ARPEGE, les logiciels LFI -**L**ogiciel de **F**ichiers (séquentiels-)**I**ndexés et FA -**F**ichiers **A**rpège(+Aladin) ont été développés pour la gestion de la plupart des fichiers de données (champs, constantes) . Les premières versions de ces logiciels datent respectivement de 1989 et 1990 ; l'utilisation en opérationnel de 1989 (avec EMERAUDE pour LFI) et 1992 .

Ces logiciels structurés en couches (FA reposant sur les services de LFI, en particulier) ont été conçus comme des éléments de bibliothèque binaire : la bibliothèque auxiliaire d'ARPEGE/ALADIN . Il s'agit donc avant tout, pour l'utilisateur, de sous-programmes FORTRAN sur lesquels on peut bâtir une application plus ou moins complexe . Les notes ARPEGE ou documentations relatives à LFI et FA ont été rédigées dans cet esprit .

Cela étant, chacun de ces logiciels a été doté d'un programme de test, dont la vocation initiale était la mise au point de chacune des parties -dans le cadre du développement initial, puis des évolutions ultérieures .

L'expérience (quelquefois fort tardive !) ayant montré que ces programmes présentaient aussi un intérêt pour l'utilisateur final, ainsi que pour des administrateurs, l'objet de cette note est donc de documenter l'utilisation de ces codes . Quitte à privilégier les exemples concrets par rapport à une exhaustivité aride.

Le principal intérêt des programmes de tests est de **pouvoir se servir du ou des logiciels, et donc des fichiers associés, sans écrire une ligne de code source**. Cela permet par exemple d'"ausculter" des fichiers particuliers (de manière générale ou fouillée), de faire du copier/coller de données, de vérifier (voire changer) la date des données ou l'identificateur dans un fichier FA ...

2) Aspects communs

2.1) Généralités

Les programmes de test des logiciels FA et LFI s'appellent respectivement TESTFA et TSTLFI (noms des points d'entrée, à la "casse" près. Leur code-source est inclus dans les mêmes branches ClearCase de la bibliothèque auxiliaire ARPEGE/ALADIN que les sous-programmes eux-mêmes (xrd -version courante à la date de rédaction de cette note: xr19, la xr20 étant bientôt sous presse). La gestion des codes et des binaires au niveau Météo France étant assurée par l'équipe GCO de SCEM/PREVI/COMPAS (adresse électronique : gco@meteo.fr) .

TESTFA et TSTLFI sont bâtis sur le même principe. Ce sont des ***interpréteurs de commandes*** (directives). Ils permettent à leur utilisateur :

- d'***invoquer toutes les primitives utilisateur*** des logiciels respectifs ;

- de ***travailler de manière interactive***, par le biais de "prompts" (impressions demandant l'action à faire, puis les arguments d'entrée), les directives étant lues (au moins initialement) sur le "standard input" du programme ;

- de ***travailler de manière programmée***, en utilisant un fichier de directives constitué avant l'exécution ;

- de ***mixer les deux modes de travail ci-dessus***, en utilisant des fichiers de directives secondaires, avec des possibilités de manipulation de ceux-ci .

A noter que TESTFA permet aussi d'invoquer une partie des primitives utilisateur LFI, en particulier les réglages des modes de fonctionnement, et les primitives sans équivalent FA (catalogue, accès pseudo-séquentiels) .

Il n'y a pas de syntaxe très structurée, type traitement conditionnel ou bouclage . Ces programmes ne sont donc pas à même de traiter de manière exhaustive tous les champs/articles d'un fichier, sauf bien sûr à spécifier explicitement ceux-ci (ou à faire un bouclage "à vue").

Le **canevas général des directives** est de la forme :

Directive_action_1 (ex: OUV pour demander l'ouverture d'un fichier)
Arguments_simples_en_entrée_de_l_action_1 (à de rares exceptions près, il y en a toutjours)
Directive_action_2
Arguments_simples_en_entrée_de_l_action_2
...
STOP (ou fin de fichier de directives)

Les **directives d'action** sont des ***noms-clés*** (chaînes de caractères, *sans délimiteurs*), qui sont en règle générale le suffixe du nom de sous-programme (sur 3 caractères pour LFI, sur 4 ou 3 caractères pour FA). Exemples : NION pour le sous-programme FA renseignant sur l'existence d'un champ et sur ses caractéristiques associées -FANION-, LAF pour le sous-programme de catalogue d’articles LFILAF . A noter l’existence de raccourcis ou alias pour FA : OUV->[FA]ITOU, FER->[FA]IRME, INI->[FA]RINE .

Les **arguments d’entrée**, qui à une exception près sont limités aux variables simples, sont à donner ***dans l’ordre où ils figurent dans la syntaxe d’appel*** du sous-programme considéré, et ***tel que suggéré par le "prompt"*** (avec les noms des variables tels qu’utilisés par le programme) . Ces arguments sont lus par un READ FORTRAN en format libre (FMT=*) ce qui conduit aux remarques suivantes :

- les **<u>variables</u> de type *caractère*** sont à ***délimiter par des ’*** (quotes simples) ;

- **on peut omettre un ou plusieurs arguments** (voire tous), ***en spécifiant /*** (slash) , ce qui a pour effet de sortir de l’instruction READ sans redéfinir la ou les variables qui n’ont finalement pas été données : le programme utilise alors les valeurs courantes des variables associées .

Cette dernière action est intéressante, par exemple, lorsqu’on veut lire un article ou un champ sans connaître forcément toutes ses caractéristiques -ou sans avoir à les saisir toutes. Pour ce faire, il suffit par exemple d’enchaîner les actions NFO et LEC pour LFI, NION et CILE pour FA; la liste des arguments d’appel de la première action respective permettant, après exécution de cette dernière, de définir les autres éléments nécessaires, pour la seconde action cela se résume à "/". C’est encore plus intéressant dans le cas d’un accès pseudo-séquentiel.

Des exemples concrets sont fournis aux chapitres suivants.

<u>2.2) Gestion du tableau de données</u>

Dans chaque programme, il y a un tableau de données :

- typé flottant (64 bits) dans TESTFA, car réputé contenir des valeurs de champs, et est dimensionné au maximum gérable par le logiciel (utilisation de facomp.h) ;

- typé entier (64 bits) dans TSTLFI.

Ce tableau est en principe destiné à être initialement le réceptacle d’une lecture, mais il y a une initialisation par défaut du type xTAB(J)=J (peu réaliste dans le cas de FA).

Outre les modifications de contenu induites par les actions "logiciel" demandées, il est aussi possible:

- d’afficher des valeurs contenues dans le tableau de données:

PRI
I1 I2 I3

et on aura sur le "standard output" du programme l’impression, en format libre (attention à la largeur de ligne sur VPP), les valeurs du tableau, au sens des boucles FORTRAN: à partir de l’indice *I1*, pas plus loin que l’indice *I2*, et par pas de *I3* valeurs.

- de spécifier tout ou partie des valeurs contenues dans le tableau de données:

TAB
I1 I2 I3

et le programme demandera, en format libre sur l'unité de lecture en cours, les valeurs du tableau, au sens des boucles FORTRAN: READ (UNIT=INPUT,FMT=*) (xTAB(J),J=*I1,I2,I3)* .

- de mettre à zéro tout ou partie des valeurs contenues dans le tableau de données:

RAZ
I1 I2 I3

Dans les trois cas ci-dessus, les valeurs spécifiées pour *I1* et *I2* (indices de début et de fin de boucle) sont écrétées aux valeurs licites (1 à JPXCHA).

2.3) Utilisation de fichiers de directives secondaires

Pour des actions répétitives, il y a intérêt à avoir des directives déjà saisies, ce qui peut se décliner de diverses manières :

a) en un fichier de directives complet, utilisé par redirection de l'entrée standard :

testfa < directives

b) en une variante du cas précédent, encapsulée dans un script pour permettre l'interprétation de variable(s) d'environnement UNIX :

tstlfi <<Fin_de_fichier
OUV
23 .T. '*$FICHIER_A_PHOTOGRAPHIER*' 'OLD' .T. .T. 1 1
LAF
23 .F.
FER
23 'KEEP'
STOP
Fin_de_fichier

l'exemple ci-dessus permet, pour un fichier LFI (et en particulier FA) de nom variable, paramétré par la variable d'environnement *FICHIER_A_PHOTOGRAPHIER*, de faire la liste des articles logiques et de leur longueur, et d'obtenir (demande à l'ouverture, mais résultat lors de la fermeture) la boîte de statistiques, dont l'intérêt dans ce contexte est de connaître des informations très générales sur le fichier et son passé .

c) de manière combinée, en "factorisant" des actions répétitives dans un voire plusieurs fichiers que l'on baptise alors **"fichiers de directives secondaires"** (vus uniquement sous la forme d'unités logiques par le programme de tests, jamais explicitement par leur nom).

Un exemple de ce genre de choses peut être donné par un fichier de type "emballage", c'est-à-dire :

- se chargeant de l'ouverture d'un fichier de nom local prédéterminé, sous une unité logique prédéterminée, puis renvoyant la main au "standard input" du programme ;

- puis, si on lui redonne la main, fermant l'unité logique courante, et renvoyant la main au "standard input" ;

- si on le rappelle une troisième fois (sans l'avoir rembobiné (cf la directive REWI ci-après)), arrête le programme.

Entre les deux premières actions ci-dessus, on peut demander toutes les actions que l'on veut. Des exemples sont donnés aux chapitres suivants.

Les **directives en jeu** pour ce genre d'actions sont :

- CHU : CHangement d'Unité logique (FORTRAN) pour la lecture des directives

- REWI : *REWIND* FORTRAN de l'unité logique considérée (contenant en principe des directives)

- CLOS : *CLOSE* FORTRAN de l'unité logique considérée (...)

Dans les trois cas l'unique argument d'appel "unité logique" est ensuite demandé.

L'unité logique FORTRAN sur laquelle on lit les directives est en effet paramétrable . A l'initialisation du programme de tests, on démarre avec l'unité 5 qui correspond par convention générale à l'entrée standard.

De manière corrélée avec ce qui précède, il faut noter que tout fichier de directives secondaires spécifié dans l'argument d'entrée correspondant à l'action CHU est ensuite référencé sans ouverture explicite, donc doit *a priori* correspondre, au moment de l'exécution, au nom de fichier par défaut associé par FORTRAN à l'unité logique en question. La nomenclature est variable selon le système (ex: fort.11 pour l'unité 11 sur VPP). Penser à un lien symbolique, voire aux possibilités locales éventuelles (exemple : via la commande "assign" sous UNICOS).

3) Exemples avec TSTLFI

3.1) Chargement de TSTLFI sur VPP

(exemple à actualiser en fonction du contexte : versions du compilateur, des bibliothèques et des outils GCO)

```
alias -x sld=~marp001/public/vpp/sld
REP_LIB=~marp001/public/bin/xrd
LIB_PROG=$REP_LIB/xr19_main.01.L9708.x.a
export SLDUSR="$REP_LIB/xr19_main.01.L9708.x.a $REP_LIB/xr19_bf.01.L9708.x.a"
rm -f sld.*                    #  attention aux effets de bord des éditions de liens antérieures (dummy)
sld -o tstlfi -e tstlfi.o -dummy
```

3.2) Catalogue d'un fichier LFI (ou FA)

Voir le paragraphe 2.3, exemple donné à l'alinéa b) .

3.3) Lecture du premier article d'un fichier LFI, impression de valeurs

```
tstlfi <<Fin_de_fichier
OUV
23 .T. '$FICHIER_A_ANALYSER' 'OLD' .T. .T. 1 1
CAS                                     # Obtention du nom et de la longueur
23 .F.                                  # du premier champ.
LEC                                     # Lecture du premier champ
/
PRI                                     # Impression de quelques valeurs
1 20 1
FER                                     # Fermeture du fichier
23 'KEEP'
STOP
Fin_de_fichier
```

3.4) Copier/coller entre 2 fichiers LFI

```
tstlfi <<Fin_de_fichier
OUV
23 .T. '$FIC1' 'OLD' .T. .T. 1 1        # Ouverture du premier fichier "FIC1"
OUV
24 .T. '$FIC2' 'OLD' .T. .T. 1 1        # Ouverture du second fichier "FIC2"
LAF                                     # Impression de la liste des articles logiques de FIC1
23 .F.
NFO                                     # Informations sur l'article de FIC1 que l'on veut
23 'S020POT.VITESSE'                    # copier sur FIC2
LEC                                     # Lecture de l'article précédent sur FIC1
/
ECR                                     # Ecriture de l'article précédent sur FIC2
24 /
```

FER
23 ’KEEP’
FER
24 ’KEEP’
STOP
Fin_de_fichier

3.5) Utilisation du mode "mise au point" du logiciel

Il suffit d’introduire la séquence suivante dans les directives à tstlfi:

DEB
.T.

4) Exemples avec TESTFA

4.1) Chargement de TESTFA sur VPP

(exemple à actualiser en fonction du contexte : versions du compilateur, des bibliothèques et des outils GCO)

```
alias -x sld=~marp001/public/vpp/sld
REP_LIB=~marp001/public/bin/xrd
LIB_PROG=$REP_LIB/xr19_main.01.L9708.x.a
export SLDUSR="$REP_LIB/xr19_main.01.L9708.x.a $REP_LIB/xr19_bf.01.L9708.x.a"
rm -f sld.*                    #  attention aux effets de bord des éditions de liens antérieures (dummy)
sld -o testfa -e testfa.o -dummy
```

4.2 Catalogue du contenu d’un fichier FA

Voir le paragraphe 2.3, alinéa b) , utilisant TSTLFI .

Variante des directives, en utilisant TESTFA (il faut ajouter un nom de cadre, dont le contenu importe peu pourvu qu’il ne dépasse pas 16 caractères et ne soit pas vide) :

testfa <<Fin_de_fichier
OUV
23 .T. ’*$FICHIER_A_PHOTOGRAPHIER*’ ’OLD’ .T. .T. 1 1 ’un_nom_de_cadre’
LAF
23 .F.
FER

23 ’KEEP’
STOP
Fin_de_fichier

N.B. : on peut aussi remplacer les "alias" OUV et FER respectivement par ITOU et IRME .

4.3 Lecture du premier champ d’un fichier FA, impression de valeurs

Cet exemple est donné dans le cadre de l’interactif; les directives tapées sont en **gras** et le prompt du logiciel en italique (*ACTION ?*). Quelques commentaires ont été ajoutés après un "#":

./testfa # L’exécutable testfa est supposé être dans le répertoire courant
ACTION ?
OUV # Le fichier ’FICHIER’ est supposé être dans le répertoire courant
INUMER, LLNOMM, CLNOMF, CLSTTU, LLERFA, LLIMST, INIMES, INBARP, CLNOMC ?
23 .T. ’FICHIER’ ’OLD’ .F. .F. 1 1 ’cadre’
///// LFIOUV - Nom=’FICHIER’

///// LFIOUV - Unite 23 OUVERTE, derniere Modification OK a 19990105_023548,
185 Articles de donnees, 3081216 mots en tout
IREP = 0
NFIOUV = 1 NULOGI = 23
ACTION ?
CAS # Obtention du nom et de la longueur du premier champ
INUMER, LLAVAN ?
23 .F.
CLNOMA = "SURFPRESSION ", ILONG = 11376 , IPOSEX = 9523
IREP = 0
NFIOUV = 1 NULOGI = 23
ACTION ?
NION # Précisions sur ce premier champ à l’aide du nom d’article
INUMER, CLPREF, INIVAU, CLSUFF ? # CLNOMA="SURFPRESSION " qu’il faut décomposer
23 ’SURF’ 0 ’PRESSION’ # en préfixe=’SURF’, numérodu niveau=0 (car absent) et
# suffixe=’PRESSION’
LLEXIS= T , LLCOSP= T ,INGRIB= 2 , INBITS= 24 ,ISTRON= 10 , IPUILA= 1
IREP = 0
NFIOUV = 1 NULOGI = 23
ACTION ?
CILE # Lecture de ce premier champ
INUMER, CLPREF, INIVAU, CLSUFF, LLCOSP ?
/
IREP = 0
NFIOUV = 1 NULOGI = 23
ACTION ?
PRI # Impression de quelques valeurs
I1, I2, I3 ?
1 8 2

```
 11.49919758763840 0.000000000000000 5.596853636029944E-03 8.869214538386279E-04
 IREP =  0
 NFIOUV =  1  NULOGI =  23
 ACTION ?
FER                                          # Fermeture du fichier
 INUMER, CLSTTU ?
23 'KEEP'
 ///// LFIFER - Unite 23 traitee, Fichier non modifie & GARDE

 ///// LFIFER - Nom='FICHIER'

 IREP =  0
 NFIOUV =  0  NULOGI =
 ACTION ?
STOP
 /// ARRET D'EXECUTION DU PROGRAMME DE TEST PAR ORDRE 'STOP'
```

4.4 Copier/coller d'un champ entre 2 fichiers FA

```
testfa <<Fin_de_fichier
OUV
23 .T. '$FIC1' 'OLD' .T. .T. 1 1 'nom_de_cadre_1'     # Ouverture du premier fichier "FIC1"
CIES                                                   # Obtention du contenu du cadre du FIC1
/
DIES                                                   # Obtention de la date des données du FIC1
/
OUV
24 .T. '$FIC2' 'OLD' .T. .T. 1 1 'nom_de_cadre_2'     # Ouverture du second fichier "FIC2"
CIES                                                   # Obtention du contenu du cadre du FIC2
/                                                      # à comparer avec celui de FIC1 pour la cohérence
DIES                                                   # Obtention de la date des données du FIC2
/                                                      # (pour contrôle éventuel)
COCH                                                   # Sélection d'un champ du FIC1 à copier sur FIC2
23 24 'S' 15 'TEMPERATURE'
FER
23 'KEEP'
FER
24 /
STOP
Fin_de_fichier
```

N.B. : on peut demander le catalogue des articles de FIC1 pour choisir l'article à recopier sur FIC2 (voir le paragraphe 4.2).

4.5 Impression de la date des données, et de l'"identificateur" du fichier FA

```
testfa <<Fin_de_fichier
OUV
23 .T. '$FICHIER' 'OLD' .T. .T. 1 1 'nom_de_cadre'  # Ouverture du fichier "FICHIER"
DIES                                               # Obtention de la date des données du fichier
/                                                    # (tableau de 11 chiffres comportant:
#  année civile, mois, jour, heure, minute du réseau de départ, 4 valeurs caractérisant l'échéance
# et 2 valeurs pour les moyennes ou les cumuls temporels (voir doc pour ces 6 dernières valeurs)
LSIF                                                 # Obtention de l'identificateur du fichier
/
FER
23 'KEEP'
STOP
Fin_de_fichier
```

4.6) Utilisation d'un fichier de directives secondaires

```
FICHIER_A_AUSCULTER=~mxpt001/arpege/t199/oper/fic_day/ICMSHARPE+0024
```

A/ Contenu du fichier "initialisation_fichier_directives_secondaires"

```
# Initialisation de variables par défaut
#
: ${STATISTIQUES=T}  ${MESSAGERIE=1}  ${UNITE_F=23}  ${UNITE_D=11}
: ${CATALOGUE_EXHAUSTIF=F} ${FICHIER_A_AUSCULTER=Non_Specifie}

if [[ "$FICHIER_A_AUSCULTER" = "Non_Specifie" ]]
then FICHIER_NOMME=F
else FICHIER_NOMME=T
fi

cat <<Fin_de_fichier > fort.$UNITE_D
OUV                                          # Ouverture du fichier FA considéré
$UNITE_F .$FICHIER_NOMME. '$FICHIER_A_AUSCULTER' 'OLD' .F. .$STATISTIQUES.
$MESSAGERIE 1 'bof'                          # Attention, une seule ligne...
CIES                                         # Obtention du contenu du cadre
/
CHU                                          # Retour à l'entrée standard
5
LAF                                          # Catalogue des articles du fichier (au sens LFI)
$UNITE_F .$CATALOGUE_EXHAUSTIF.
CHU                                          # Retour à l'entrée standard
5
FER                                          # Fermeture du fichier
$UNITE_F 'KEEP'
CHU                                          # Retour à l'entrée standard
5
```

STOP # Arrêt du programme
Fin_de_fichier

B/ Exécution en interactif de testfa:

Le fichier fort.11 obtenu après l'exécution de "initialisation_fichier_directives_secondaires"
est supposé être dans le répertoire courant.

./testfa # Supposé être dans le répertoire courant
ACTION ?
CHU
NOUVELLE UNITE LOGIQUE D'INPUT ?
11 # On donne la main aux directives secondaires

...
(Ouverture du fichier FA, visualisation du cadre ...)
...

Puis le fichier de directives secondaires redonne la main pour l'interactif:

ACTION ?
DIES # Obtention de la date des données
INUMER ?
/
IDATEF= 1998 11 3 12 0 1 0 0 0 0 0
IREP = 0
NFIOUV = 1 NULOGI = 1
ACTION ?
CHU
NOUVELLE UNITE LOGIQUE D'INPUT ?
11 # On donne la main aux directives secondaires

...
Catalogue des articles du fichier (au sens LFI)
...

Puis le fichier de directives secondaires redonne la main pour l'interactif:

ACTION ?
NION # Existence/caractéristiques d'un champ
INUMER, CLPREF, INIVAU, CLSUFF ?
23 'S' 15 'TEMPERATURE'
LLEXIS= T , LLCOSP= T ,INGRIB= 2 , INBITS= 24 ,ISTRON= 10 , IPUILA= 1
IREP = 0
NFIOUV = 1 NULOGI = 1
ACTION ?
CILE # Lecture du champ considéré (spectral ou pdg)
INUMER, CLPREF, INIVAU, CLSUFF, LLCOSP ?

/
 IREP = 0
 NFIOUV = 1 NULOGI = 1
 ACTION ?
PRI # Impression de quelques valeurs du champ décodé
 I1, I2, I3 ?
1 16 1
 254.1282428531473 -0.86413699594906 -2.58118922384459 -5.84070951554544 -0.49987534062992
0.460679691197213 -0.19185660003298 -0.17310513271999 -1.57453589496837 -
7.009681121079418E-03 -0.12607239559639 0.692962973480324 6.286362752171347E-02
0.621454821592525 0.858000894946952 -0.13877289858412
 IREP = 0
 NFIOUV = 1 NULOGI = 1
 ACTION ?
CHU
 NOUVELLE UNITE LOGIQUE D'INPUT ?
11 # On donne la main aux directives secondaires

...
(Fermeture du fichier ...)
...

Puis le fichier de directives secondaires redonne la main pour l'interactif:

 ACTION ?
LIMU # Obtention des limites utilisateur en résolution
 IXNIVV = 120 , IXTRON = 319 , IXLATI = 400 , IXLONG = 800
 IREP = 0 # en l'absence de réglage par (FA)RFLU, ce sont les limites
 NFIOUV = 1 NULOGI = 1 # logiciel (PARAMETER)
 ACTION ?
CHU
 NOUVELLE UNITE LOGIQUE D'INPUT ?
11 # On donne la main aux directives secondaires
qui arrête l'exécution du programme testfa.

IREP = 0
 NFIOUV = 1 NULOGI = 1
  ACTION ?
/// ARRET D'EXECUTION DU PROGRAMME DE TEST PAR ORDRE 'STOP'

4.7 Utilisation d'une messagerie bavarde pour FA, mais pas pour LFI

Il suffit d'introduire la séquence suivante (attention à l'ordre) dans les directives à testfa:

NMSG
2
NMG
1

4.8 Utilisation du mode "mise au point" du logiciel

Il suffit d'introduire la séquence suivante dans les directives à testfa:

DEB
.T.