Les librairies en Fortran

Module d'informatique - Licence DSM ENS Lyon

TD n° 8 — 2002-2003

26 novembre 2002, y.copin@ipnl.in2p3.fr

Objectifs du TD : Compilation et édition de liens, réalisation et utilisation de librairies

1 Introduction

L'objectif de *modularité* est primordial pour une programmation efficace. En effet, comme dans de nombreux domaines, un certain nombre d'opérations de base sont communes à de très nombreux calculs, et il serait particulièrement vain de les redéfinir à chaque nouvelle utilisation. Il est donc important :

- de bien identifier ces fonctions élémentaires,
- de les isoler au mieux de leur environnement,
- de les optimiser et éventuellement d'accroître leur potentiel en généralisant leurs domaines d'application.

Ainsi, comme dans un jeu de construction, il suffira alors de les assembler, ou même simplement de les enrober, pour créer rapidement des programmes plus complexes.

En ce sens, le FORTRAN fournit les `function`s et `subroutine`s, pour lesquelles il est important :

- de circonscrire l'action : chaque routine doit avoir un unique objectif bien défini, et déporter tout calcul complexe à une sous-routine (et ainsi de suite) ;
- de définir précisément son mode d'emploi, en particulier en précisant clairement l'interface (c-à-d la nature et le rôle des différents paramètres ou variables globales), les conditions d'utilisation (p.ex. tailles maximale des tableaux), les codes d'erreur, etc.

Cette démarche doit se faire non seulement au niveau du code lui-même (p.ex. par l'utilisation de noms de variable explicites), mais également au niveau de la documentation détaillée des procédures.

Une fois ce travail de normalisation et documentation réalisé, il est intéressant de regrouper l'ensemble de ces fonctions de base au sein d'une *bibliothèque* sur laquelle se basera les développements futurs. Ainsi, il existe de par le WEB de nombreuses bibliothèques FORTRAN couvrant de multiples domaines (p.ex. calculs numériques, fonctions mathématiques, entrées-sorties, etc.) dont l'utili-

sation permet non seulement d'accélérer le développement de son projet, mais aussi de contribuer à la structuration de son code.

2 Compilation et édition de liens

Le terme générique de « *compilation* », utilisé jusqu'à présent pour évoquer la production d'un programme exécutable à partir d'un fichier source, recouvre en fait deux actions distinctes :

La compilation à proprement parler traduit un fichier *source* (p.ex. d'extension `.f`), crée à l'aide d'un éditeur de texte, en un fichier *objet* (extension `.o`) : il s'agit simplement de la traduction en langage machine de la suite d'instructions du fichier source. Chaque donnée ou fonction définie dans un fichier objet possède un *nom symbolique*, et chaque référence à un symbole extérieur au fichier objet considéré est appelée *référence externe*.

> **Attention :** Les fichiers objets d'extension `.o` sont des fichiers binaires mais ne sont pas exécutables.

L'édition de liens (*to link*) tente de faire le lien entre les symboles définis ou requis dans les différents fichiers objets `.o` (ou archives `.a`) pour enfin produire le fichier exécutable. Si l'éditeur de liens ne peut pas lier une référence externe, il affiche un message d'erreur et ne peut pas produire l'exécutable.

Exercice : Les différents tests de compilation se feront sur les fichiers `newton.f` et `zero.f` de la correction du TD n° 5 (cf. `http://snovae.in2p3.fr/ycopin/enseignements/info_ENS.html`) : le fichier `zero.f` contient le programme principal, ainsi que la fonction `f` à étudier et sa dérivée `df`, tandis que le fichier `newton.f` contient la définition de la fonction `newton` et de ses dépendances.

Dans les cas simples – p.ex. production d'un programme à partir d'un ou plusieurs fichiers sources –, ces deux actions sont regroupées de façon transparente dans une seule commande. Dans notre cas :

```
g77 -Wall -o zero.exe zero.f newton.f
```

produit l'exécutable `zero.exe` à partir des deux fichiers `zero.f` et `newton.f`, comme si ces deux fichiers avaient été concaténés en un seul (avec l'avantage cependant de la modularité).

Cependant, dans les cas plus complexes[1], les deux actions doivent être séparées.

[1] Même si ici, cela ne s'avère évidemment pas indispensable...

2.1 Compilation

La production d'un fichier objet `.o` à partir d'un fichier source `.f` se fait dans notre cas à l'aide de l'option `-c` de la commande `g77`. Ainsi, les commandes

```
g77 -Wall -c newton.f
g77 -Wall -c zero.f
```

créent-elles les fichiers `newton.o` et `zero.o`.

2.2 Édition de liens

L'éditeur de liens utilise la table des symboles des différents fichiers objets pour assortir les références externes (c-à-d les symboles utilisés mais non *définis* dans un fichier objet) aux définitions globales (les symboles définis dans un fichier objet). Sous UNIX, l'édition de liens se fait à l'aide de la commande `ld` *quelque soit le langage de programmation*. Cependant, on n'utilise rarement cette commande directement, puisque la plupart des compilateurs offre une interface plus simple d'emploi. Dans notre cas, il s'agit simplement de la commande `g77`. P.ex.

```
g77 -o zero.exe newton.o zero.o
```

génère le programme exécutable `zero.exe` à partir des fichiers objets `newton.o` et `zero.o`.

3 Les librairies

Les *librairies* – ou *bibliothèques* – ne sont qu'un type particulier de fichier objet, regroupant dans un même fichier des fonctions de base très souvent utilisées. Tout langage de programmation dispose de ses propres librairies, et il est souvent intéressant de se constituer des librairies personnelles regroupant des routines autour d'une thématique précise. C'est également ce que l'on peut trouver sur le réseau Internet, offrant de multiples fonctionnalités dans de nombreux domaines (p.ex. calculs numériques, gestion des entrées-sorties, etc.).

> **Attention :** Une librairie doit être parfaitement structurée et documentée pour pouvoir être utile à terme (c-à-d au delà de l'utilisation immédiate).

Pour ce faire, il est préférable de séparer les différentes routines d'une librairie en autant de fichiers – contenant chacun une unique fonctionnalité et ses dépendances –, et de normaliser l'entête de ce fichier pour indiquer clairement les informations utiles. P.ex., le fichier `binom.f` de la librairie SLATEC (voir ci-dessous) utilise l'entête suivant :

```
      FUNCTION BINOM (N, M)
C***BEGIN PROLOGUE  BINOM
C***PURPOSE  Compute the binomial coefficients.
C***LIBRARY   SLATEC (FNLIB)
C***CATEGORY  C1
C***TYPE      SINGLE PRECISION (BINOM-S, DBINOM-D)
C***KEYWORDS  BINOMIAL COEFFICIENTS, FNLIB, SPECIAL FUNCTIONS
C***AUTHOR  Fullerton, W., (LANL)
C***DESCRIPTION
C
C BINOM(N,M) calculates the binomial coefficient (N!)/((M!)*(N-M)!).
C
C***REFERENCES  (NONE)
C***ROUTINES CALLED  ALNREL, R1MACH, R9LGMC, XERMSG
C***REVISION HISTORY  (YYMMDD)
C   770701  DATE WRITTEN
C   890531  Changed all specific intrinsics to generic.  (WRB)
C   890531  REVISION DATE from Version 3.2
C   891214  Prologue converted to Version 4.0 format.  (BAB)
C   900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
C   900326  Removed duplicate information from DESCRIPTION section.
C           (WRB)
C***END PROLOGUE  BINOM
```

Attention : La librairie SLATEC utilise les variables implicites, ce qui la dispense d'expliciter le type des variables d'entrée. Dans votre cas, il faut *en plus* décrire la nature des paramètres de la routine.

3.1 Création

Par convention, le nom d'une librairie (statique, par opposition aux librairies partagées que l'on n'abordera pas ici.) est `lib`*`nomdelalibrairie`*`.a`. Une librairie se créé à partir d'une collection de fichiers `.o` à l'aide de la commande `ar -r`. Ainsi, dans notre cas simple :

```
ar -r libnewton.a newton.o
```

créé une librairie `libnewton.a` contenant seulement le fichier `newton.o`.

Attention : En règle générale, on inclut dans une archive des fichiers objets ne contenant *que* des fonctions ou sous-routines, mais pas de programmes principaux.

3.2 Utilisation

Une librairie n'intervient bien entendu qu'au niveau de l'édition de lien. Elle peut alors être utilisée comme n'importe quel fichier objet, p.ex.

```
g77 -o zero.exe zero.o libnewton.a
```

ou même plus directement

```
g77 -o zero.exe zero.f libnewton.a
```

cette dernière commande incluant la compilation « à la volée » de `zero.f`.

Lorsque la librairie est stockée dans un répertoire différent du répertoire courant (p.ex. dans le répertoire de compilation de la bibliothèque, ou dans un répertoire dédié à toutes vos bibliothèques), on peut utiliser les options de compilation `-L`*`répertoire`* `-l`*`nomlib`*, pour une librairie `lib`*`nomlib`*`.a` stockée dans le répertoire *`répertoire`*.

Exercice : Créer une librairie personnelle (p.ex. `libperso.a`) contenant les différentes routines développées dans les TD précédents, et reformuler les programmes « principaux » pour utiliser cette librairie.

4 Librairies externes

Nous vous fournissons pour ce TD deux librairies externes en F77 que vous pouvez maintenant utiliser pour le développement de vos programmes :

- la librairie pédagogique de *Numerical Recipes*,
- la librairie SLATEC.

Ces librairies sont disponibles sous `/home/ycopin/Fortran/lib`, et se nomment respectivement `libnr77.a` et `libslatec.a`. Vous pouvez éventuellement copier ces fichiers chez vous, mais il est préférable de lier vos programmes directement avec les librairies originelles.

4.1 Numerical Recipes

La NUMREC (`www.nr.com`) est une librairie « pédagogique » regroupant les différentes routines introduites et discutées dans le célèbre livre *Numerical Recipes*, disponible en ligne à `http://www.library.cornell.edu/nr/bookfpdf.html`. Il ne s'agit donc pas réellement d'une librairie à usage professionnel (elle est même réputée assez peu fiable), mais plutôt d'un bon point de départ.

> **Attention :** Les sources de la NUMREC ne sont pas gratuites ! Vous ne devez donc pas distribuer la librairie que l'on met à votre disposition.

Exercice : Utiliser la routine `mppi` de la NUMREC (§ 20.6) pour afficher un nombre arbitraire de décimales de π.

4.2 SLATEC

SLATEC (http://www.netlib.org/slatec/) est une librairie numérique d'utilisation aisée contenant plus de 1400 routines mathématiques et statistiques. Elle est de plus indexée dans le *Guide to Available Mathematical Software* (GAMS), ce qui en facilite son usage (voir p.ex. le *Problem Decision Tree*, http://gams.nist.gov/serve.cgi).

Par ailleurs, pour faciliter l'interface avec vos programmes, chacune des routines dispose de son propre manuel d'utilisation sous `/home/ycopin/Fortran/slatec/man`, consultable avec la commande `man`, p.ex.

```
man /home/ycopin/Fortran/slatec/man/binom.1
```

Exercice : Trouver dans la librairie SLATEC la routine permettant de calculer les racines d'un polynôme à coefficients réels, et l'intégrer dans un programme. Comparer plusieurs librairies (résultats, performances, facilité d'utilisation).