Création et utilisation de bibliothèques

Philippe Collet
2007-2008

D'après un cours de L. Pierre

Plan

- Motivations
- Bibliothèques statiques
- Bibliothèques dynamiques
- Utilisation combinée

Motivations et principes

- Une bibliothèque est une archive de fichiers objets.
- Une application peut ainsi utiliser des fonctions pré-programmées, sans avoir à les recompiler à chaque fois que le programme est compilé.
- La bibliothèque pourra être liée statiquement ou dynamiquement au code objet du programme.

Création d'une bibliothèque statique

```c
/* convert1.c - conversion francs-euro */
#include <stdio.h>

/* x francs = x/6.55957 euros */
void franc_euro(double franc) {
  printf("\t%.4lf francs = %.4lf euros\n",
         franc, franc / 6.55957);
}
```

```c
/* convert2.c - conversion euro-francs */
#include <stdio.h>

/* x francs = x*6.55957 euros */
void euro_franc(double euro) {
  printf("\t%.4lf euros = %.4lf francs\n",
         euro, euro * 6.55957);
}
```

```
matrix% gcc -c convert1.c
matrix% gcc -c convert2.c
```

```
matrix% ar r libconvert.a convert1.o convert2.o
matrix% ranlib libconvert.a
```

Bibliothèque statique

- Définition
  - C'est un *fichier archive* qui contient du code exécutable
  - Il débute par une *table des symboles*, i.e. une description des modules et identificateurs permettant à l'éditeur de liens de résoudre les références correspondantes sans avoir à parcourir toute la bibliothèque.
- Forme
  - Les bibliothèques statiques ont un nom de la forme lib*nom*.a ; elles peuvent contenir un ou plusieurs fichiers objets.
- Création
  - grâce à la commande ar (l'option r est pour l'insertion).
  - Pour générer l'index de l'archive et le stocker dans l'archive, on utilise ensuite la commande ranlib.

Consulter l'archive

- l'option t de ar permet de *consulter la liste des fichiers* objets contenus dans l'archive

```
matrix% ar t libconvert.a
convert1.o
convert2.o
```

- la commande nm permet d'obtenir la *liste des symboles* se trouvant dans des *fichiers objets*

```
matrix% nm -s libconvert.a

libconvert.a[conv1.o]:
[Index]  Value   Size   Type  Bind  Other Shname     Name
[2]     |      0|      0|SECT |LOCL |0    |.rodata   |
[4]     |      0|      0|SECT |LOCL |0    |.text     |
[1]     |      0|      0|FILE |LOCL |0    |ABS       |conv1.c
[6]     |      0|     72|FUNC |GLOB |0    |.text     |franc_euro
[3]     |      0|      0|NOTY |LOCL |0    |.text     |gcc2_compiled.
[5]     |      0|      0|NOTY |GLOB |0    |UNDEF     |printf

libconvert.a[conv2.o]:
[Index]  Value   Size   Type  Bind  Other Shname     Name
[2]     |      0|      0|SECT |LOCL |0    |.rodata   |
[4]     |      0|      0|SECT |LOCL |0    |.text     |
[1]     |      0|      0|FILE |LOCL |0    |ABS       |conv2.c
[6]     |      0|     72|FUNC |GLOB |0    |.text     |euro_franc
[3]     |      0|      0|NOTY |LOCL |0    |.text     |gcc2_compiled.
[5]     |      0|      0|NOTY |GLOB |0    |UNDEF     |printf
```

Déploiement

- L'archive + Un fichier d'en-tête

```c
/* fichier convert.h */
#define EURO 0
#define FRANC 1
void franc_euro(double franc);
void euro_franc(double euro);
```

- Inclure le fichier d'en-tête dans le source
- Référencer la bibliothèque lors de la compilation pour l'édition de liens
- Les fichiers d'inclusion et les bibliothèques peuvent être placés dans des répertoires...

Utilisation

Répertoire ou chercher la bibliothèque

Nom abrégé de la bibliothèque
convert → libconvert.a

```
matrix% gcc convertir.c -L . -lconvert -o convertir
matrix% ./convertir
entrez un nombre : 100
convertir en euros (0) / francs (1) ? 0
         100.0000 francs = 15.2449 euros
entrez un nombre : 15
convertir en euros (0) / francs (1) ? 1
         15.0000 euros = 98.3936 francs
entrez un nombre : 0
matrix%
```

Utilisation (avec répertoire)

- Mettre en place
  - un répertoire lib — pour les bibliothèques
  - un répertoire include — pour les fichiers en-tête

```
matrix% gcc convertir.c -I $HOME/include \
          -L $HOME/lib -lconvert -o convertir
```

- Recherche sur #include <fic.h>
  - les répertoires spécifiés par l'option -I
  - le(s) répertoire(s) standard(s) pour les fichiers d'en-tête, classiquement /usr/include
- Recherche sur #include "fic.h" (sauf en cas de chemin absolu)
  - le répertoire courant
  - les répertoires spécifiés par l'option -I
  - le(s) répertoire(s) standard(s) pour les fichiers d'en-tête, classiquement /usr/include

Principe d'une bibliothèque dynamique

- Compiler les fichiers de façon à **générer des fichiers objets relogeables**
- Créer la bibliothèque spécifique
- Un chargeur va
  - réaliser le chargement dynamique de la bibliothèque
  - rendre le programme réellement exécutable (puis l'exécuter)
  - Le code contenu dans la bibliothèque doit pouvoir être déplacé en mémoire, c'est pourquoi il est qualifié de relogeable
- Fonctionnement d'un code relogeable
  - toute adresse assortie d'un "relocation bit" (placé lors de la génération du code objet) est en fait une adresse relative à un segment
  - l'adresse absolue qui lui correspondra dans le code exécutable est calculée comme un déplacement de cette adresse relative par rapport à une adresse de base.

Création d'une bibliothèque dynamique

- Compiler les fichiers au format PIC
  - PIC = *Position Independent Code*
  - option **-fPIC**.

```
matrix% gcc -fPIC -c convert1.c convert2.c -I $HOME/include
```

- Créer la bibliothèque
  - option **–shared**

```
matrix% gcc -shared -o libconvert.so convert1.o convert2.o
```


Utilisation

```
matrix% gcc convertir.c -I $HOME/include \
            -L $HOME/lib -lconvert -o convertir
matrix% ./convertir
ld.so.1: ./convertir: fatal: libconvert.so: open failed:
                              No such file or directory
```

Pourquoi ça ne marche pas ?

```
matrix% ldd convertir
        libconvert.so =>  (file not found)
        libc.so.6 =>  /usr/lib/libc.so.6.1
        libdl.so.1 =>    /usr/lib/libdl.so.1
```

Recherche par le chargeur ld.so

- ld.so cherche les bibliothèques partagées dans les répertoires :
  - /lib et /usr/lib
  - ceux définis dans le fichier /etc/ld.so.conf
  - ceux de la variable **LD_LIBRARY_PATH**

```
matrix% setenv LD_LIBRARY_PATH ${HOME}/lib:${LD_LIBRARY_PATH}
matrix% ldd convertir
        libconvert.so =>   /u/profs/collet/lib/libconvert.so
        libc.so.1 =>     /usr/lib/libc.so.1
        libdl.so.1 =>    /usr/lib/libdl.so.1
matrix% ./convertir
entrez un nombre : ........
```


Autre possibilité

- l'option de compilation -Wl,*option* sert à passer une (ou des) options à l'éditeur de liens.
  - -Wl,-rpath
  - -rpath *dir* ajoute le répertoire *dir* à la liste du chemin de recherche des bibliothèques

```
matrix% gcc convertir.c -I $HOME/include -L $HOME/lib \
        -lconvert -Wl,-rpath,/u/profs/collet/lib/ -o convertir
matrix% ldd convertir
         libconvert.so =>  /u/profs/collet/lib/libconvert.so
         libc.so.6 =>  /usr/lib/libc.so.6.1
         libdl.so.1 =>    /usr/lib/libdl.so.1
```


Versions de bibliothèques partagées

- Motivations
  - Gérer l'évolution des bibliothèques
  - Gérer la présence simultanée de différentes versions
- Unix permet à **plusieurs versions de bibliothèques partagées** d'être présentes simultanément. Dans ce cas, la bibliothèque a un :
  - nom d'objet partagé ou *soname* : lib*Nom_bib*.so.majeur
    - majeur étant le numéro de version majeure.
    - Le *soname* est souvent un lien symbolique vers le nom réel de la bibliothèque.
    - c'est le *soname* qui est stocké dans l'exécutable.
  - nom réel : lib*Nom_bib*.so.majeur.mineur.version
    - Ce nom est généralement le *soname* auquel on a ajouté un nombre mineur suivi d'un numéro de version
  - nom de lien (nom que le compilateur utilise à l'édition des liens)
    - *soname* sans numéro de version

Exemples

```
Exemple sur un PC sous Linux :
lrwxrwxrwx   1 root      19 nov 12 10:32 /usr/lib/libm.so -> ../../lib/libm.so.6*
lrwxrwxrwx   1 root      13 nov 12 10:28 /lib/libm.so.6 -> libm-2.2.4.so*
-rwxr-xr-x   1 root  139200 sep  4 20:31 /lib/libm-2.2.4.so*

Exemple sur un SUN :
-rw-r--r--   1 bin   1607728 Aug 30  1999 libc.a
lrwxrwxrwx   1 root       11 Jan 24  2001 libc.so -> ./libc.so.1
-rwxr-xr-x   1 bin   1014012 Aug 30  1999 libc.so.1
-rw-r--r--   1 bin    146708 Aug 22  1996 libm.a
lrwxrwxrwx   1 root        9 Jan 24  2001 libm.so -> libm.so.1
-rwxr-xr-x   1 bin    105788 Aug 22  1996 libm.so.1
```

Règles de numérotation

- La *première version* a le numéro 1.0 ou 1.0.1 (major.minor.release)
- Si la modification est *rétro-compatible*, on augmente le *numéro de version mineure*.
  - nouvelle version avec des améliorations mais qui ne rajoute pas de nouvelles fonctions, les modifications apportées n'affectant que le fonctionnement interne de la bibliothèque et le nombre de fonctions et de variables, les paramètres et types de valeurs renvoyées pour les fonctions restant intacts.
- Si la modification est *incompatible avec les versions antérieures*, on augmente le *numéro de version majeure*.
  - suppression ou ajout de fonctions, modification de syntaxes d'appel de fonctions,...
- Pour une version majeure ou mineure, une *augmentation du dernier chiffre* signifie des corrections de bogues, une compatibilité conservée et aucun changement significatif.

Mise en œuvre

```
matrix% gcc -fPIC -c convert1.c convert2.c -I $HOME/include
matrix% gcc -shared -Wl,-soname,libconvert.so.1 \
                    -o libconvert.so.1.0.1 convert1.o convert2.o
matrix% ln -s libconvert.so.1.0.1 libconvert.so.1
matrix% ln -s libconvert.so.1 libconvert.so
matrix% ls -l
..........
lrwxrwxrwx  1 collet     15 mar 13 11:42 libconvert.so -> libconvert.so.1*
lrwxrwxrwx  1 collet     19 mar 13 11:38 libconvert.so.1 -> libconvert.so.1.0.1*
-rwxr-xr-x  1 collet   5869 mar 13 10:51 libconvert.so.1.0.1*
matrix% gcc convertir.c -I /u/profs/collet/include -L . \
                        -Wl,-rpath,/u/profs/collet/lib -lconvert \
                        -o convertir
matrix% ldd convertir
        libconvert.so.1 => /u/profs/collet/lib/libconvert.so.1
        libc.so.6 =>  /lib/libc.so.6
        libdl.so.1 =>    /usr/lib/libdl.so.1
```

Le *lien symbolique* libconvert.so est utile, car c'est un fichier de ce nom que cherche ld.so.
Le *soname* libconvert.so.1 étant le nom stocké dans le programme compilé, le lien symbolique correspondant est aussi nécessaire, c'est lui qui conduit à libconvert.so.1.0.1.

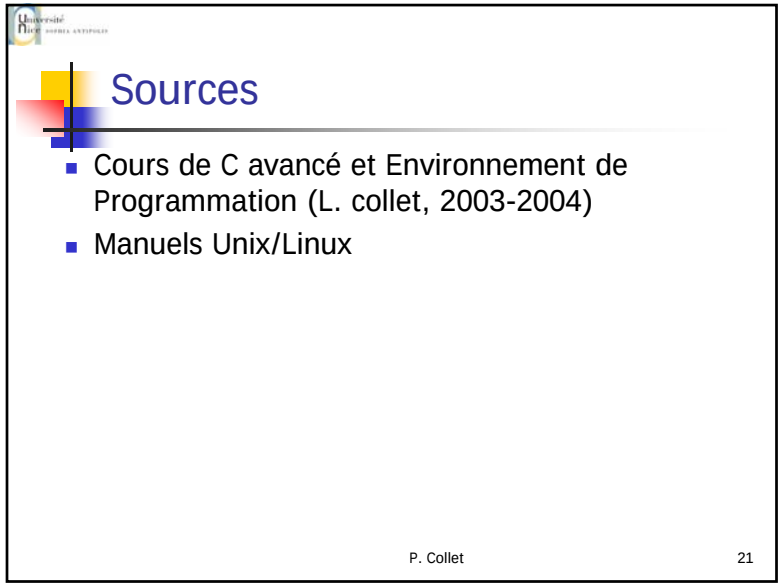
Installation de bibliothèques

- En tant que root, on peut installer de nouvelles bibliothèques dans /usr ou /usr/lib
  - `ldconfig` *crée les liens symboliques* nécessaires à la mise en place d'une nouvelle bibliothèque, et *met à jour le contenu du cache* `/etc/ld.so.cache` (ce cache peut être consulté avec `ldconfig -p`).
- Fonctionnement de `ldconfig`
  - `ldconfig` lit le fichier `/etc/ld.so.conf` et écrit un cache `/etc/ld.so.cache` qui est utilisé par les programmes afin d'accélérer l'accès aux bibliothèques.
  - ☞ lancer `ldconfig` à chaque fois qu'on installe une bibliothèque.

Usage simultané de bibliothèques statiques et partagées

- Par défaut, utilisation de la bibliothèque partagée, si elle est présente
- Pour combiner les deux, il faut stipuler le mode de reliure pour chaque bibliothèque...

```
matrix% gcc convertir.c -I $HOME/include -L . -Wl,-Bstatic \
                                        -lconvert -o convertir
matrix% ldd convertir
        not a dynamic executable
matrix%  gcc convertir.c -I $HOME/include -L . -Wl,-Bstatic \
                         -lconvert -Wl,-Bdynamic -lc -o convertir
matrix% ldd convertir
        libc.so.6 =>  /usr/lib/libc.so.6.1
        libdl.so.1 =>    /usr/lib/libdl.so.1
```


Sources
Cours de C avancé et Environnement de Programmation (L. collet, 2003-2004)
Manuels Unix/Linux
P. Collet
21