5

Desenvolvimento de Bibliotecas

Sumário:

- **Tipos de bibliotecas: arquivo (.a) e partilhada (.so)**Bibliotecas do sistema (libc, libm, ...)

- **Criação duma biblioteca estática**
- Visualização do conteúdo duma biblioteca estática
- Substituição dos módulos objecto duma biblioteca estática
- Adição de módulos objecto a uma biblioteca estática
- Eliminação de módulos objecto duma biblioteca estática
- Sumário das chaves do comando ar
- Comunicação entre funções

- **Onde colocar as bibliotecas estáticas?**
- **Unificação de bibliotecas e programas**
- Gestor de compilação de programas (make)

Tipos de bibliotecas

Há dois tipos básicos de bibliotecas:

- **Estáticas** (ou *archive*). A característica fundamental duma biblioteca estática ou arquivo é que na unificação de código objecto com um programa do utilizador, o código da biblioteca é incluído dentro do executável. Portanto, se houver cinco programas que incluam a mesma biblioteca estática no seu código executável, temos cinco cópias da biblioteca em memória, o que é claramente um desperdício em termos de ocupação da memória.
- **Dinâmicas** (ou partilhadas). Estas bibliotecas, tal como o nome no-lo diz, são partilhadas pelos programas que as usam. Só existe uma cópia duma biblioteca dinâmica em memória, independentemente do número (um ou mais) de programas que a usem em tempo de execução.

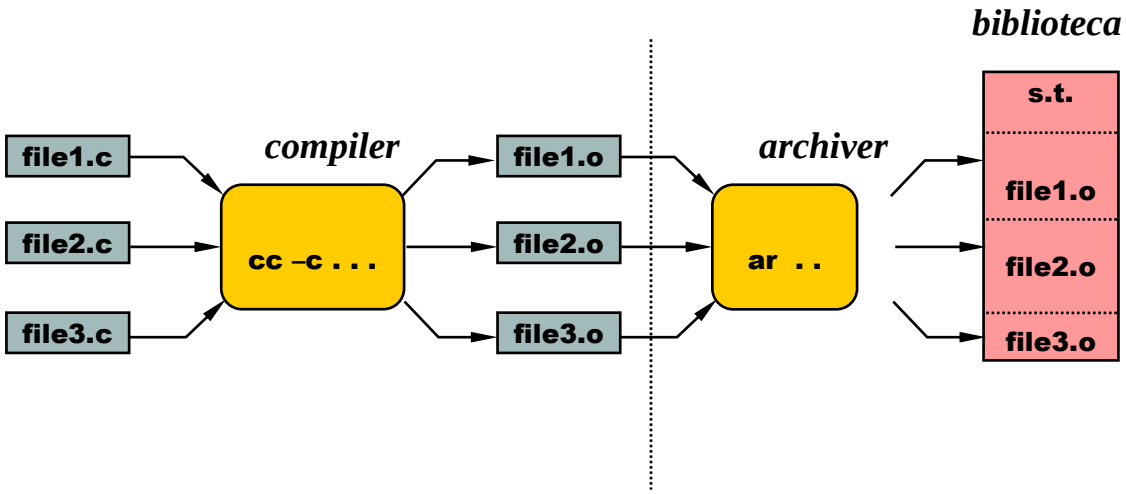
Para LINUX
As bibliotecas estáticas têm extensão **.a**, ao passo que as bibliotecas dinâmicas têm extensão **.so**.

Criação duma biblioteca estática

Há duas fases principais na criação duma biblioteca estática:

- ***Criação dos ficheiros em código objecto (*.o)***
  Nesta fase usa-se o compilador `cc` com a opção `-c`.

- ***Junção dos ficheiros em código objecto numa biblioteca estática (archive library)***
  Nesta fase usa-se o arquivador **ar**.



Conteúdo duma biblioteca estática

Uma biblioteca estática está dividida em três secções:

- ***Cabeçalho***: "!<arch>\n"
  Identifica o ficheiro referente à biblioteca como um ficheiro arquivador (*archive file*) criado por **ar** (\n representa um carácter *newline*)
- ***Tabela de símbolos***:
  É usada pelo unificador (ou *linker*) para encontrar a localização, o tamanho e ainda informação adicional relativa a cada rotina e a cada dado existentes na biblioteca.
- ***Módulos objecto***:
  Existe um para cada ficheiro objecto especificado no comando **ar**.

Exemplo de construção duma biblioteca estática: **libunits.a**

Pretende-se construir uma biblioteca de funções que estão contidas em três ficheiros distintos, nomeadamente: **length.c**, **volume.c** e **mass.c**:

length.c

```
#include <units.h>

const float VALOR=2.5;

float in_to_cm (float in)   // inches to cm
{
   return (in * VALOR);
}
```

volume.c

```
float gal_to_l (float gal)   // gallons to liters
{
   return (gal * 3.79);
}
```

mass.c

```
float oz_to_g (float oz)   // ounces to grams
{
   return (oz * 28.35);
}
```

- Geração dos ficheiros objecto
  $ cc –c length.c volume.c mass.c
- Arquivação dos ficheiros objecto na biblioteca libunits.a
  $ ar r libunits.a length.o volume.o mass.o

Criação do ficheiro de protótipos **units.h** da biblioteca libunits.a

Para se poder usar a biblioteca libunits.a, há que criar um ficheiro de protótipos das funções implementadas pela biblioteca. Seja **units.h** o ficheiro de protótipos da biblioteca **libunits.a**.

units.h

```
float in_to_cm (float);
float gal_to_l (float);
float oz_to_g (float);
```

A utilização de qualquer função desta biblioteca num programa requer a inclusão do ficheiro units.h no início do dito programa através da directiva #include. Só assim se satisfaz o princípio fundamental das linguagens imperativas: **só se pode usar uma variável (ou função) se tiver sido declarada previamente**.

Colocação da biblioteca libunits.a no sistema de ficheiros do UNIX

Após a criação da biblioteca **libunits.a**, há que guardá-la num local facilmente acessível a outros programadores. Existem dois locais principais que o unificador (ou linker) procura por defeito na fase de junção ou unificação do código objecto:

- /lib
- /usr/lib

Colocação do ficheiro de protótipos units.h no sistema de ficheiros do UNIX

Após a criação do ficheiro **units.h**, há que guardá-lo num local facilmente acessível a outros programadores:

- /include
- /usr/include

Deste modo, qualquer programa que pretenda usar a biblioteca libunits.a, deve fazer a inclusão do ficheiro de protótipos units.h do seguinte modo:

#include <units.h>

Se, porventura, o programador não colocou o ficheiro units.h numa das duas directorias estandardizadas acima indicadas, mas o colocou na directoria corrente de trabalho, a sua inclusão é feita do seguinte modo:

#include "units.h"

Se, o programador colocou na directória /home/users/projects/include a sua inclusão seria feita com

#include "/home/users/projects/include"

Exemplo de compilação sem biblioteca:

Pretende-se criar um ficheiro executável **convert.exe**, cujo código fonte é o seguinte:

convert.c

```
#include <stdio.h>
#include "units.h"
int main(void)
{
   float inches, centimetros;
   printf ( "NORMAL Escreva um valor polegadas:" );
   scanf("%f", &inches);
   centimetros = in_to_cm(inches);
   printf("O seu valor em centimetros=%f\n",centimetros);
   return(0);
}
```

- Geração do executável convert.exe **$ cc –o convert.exe convert.c length.o mass.o volume.o**

Exemplo de utilização duma biblioteca estática: **libunits.a**

Pretende-se utilizar a biblioteca libunits.a na criação dum ficheiro executável **convertst.exe**, cujo código fonte é igual em tudo do ficheiro convert.sh excepto no primeiro printf:

convertst.

```
#include <stdio.h>
#include "units.h"
int main(void)
{
   float inches, centimetros;
   printf ( "STATIC Escreva um valor polegadas:" );
   scanf("%f", &inches);
   centimetros = in_to_cm(inches);
   printf("O seu valor em centimetros=%f\n",centimetros);
   return(0);
}
```

- Geração do executável convertsh.exe **$ cc –o convertst.exe convertst.c libunits.a**

Exercício: Alterar o constante no ficheiro length.c para o valor 2.54,actualize a biblioteca estática libunits.a e criar de novo o executável. Para saber substituir uma parte duma biblioteca estática consulte o "man ar".

Criação duma biblioteca dinâmica ou partilhada

Há duas fases principais:

- ***Criação dos ficheiros em código objecto (*.o)***
  Nesta fase usa-se o compilador `cc` com a opção `–c`.
- ***Junção dos ficheiros em código objecto numa biblioteca partilhada (shared object library or dynamic link library)***
  Nesta fase usa-se o coompilador **cc (Linux)** ou libtool **(Mac/Darwin)**.

```
cc –shared -fPIC  -o nome_biblioteca ficheiros_objectos
```

Exemplo de construção duma biblioteca estática: **libunits.so**

Utilzamos o comando

$ cc – shared –fPIC –o libunits.so length.o volume.o mass.o

Nota: na maquina alunos.di.ubi.pt (Mac/Darwin) utilize-se o "libtool"

Exemplo de utilização duma biblioteca dinâmica: **libunits.so**

Pretende-se utilizar a biblioteca libunits.a na criação dum ficheiro executável **convertsh.exe**, cujo código fonte é igual em tudo do ficheiro convert.sh excepto no primeiro printf:

convertsh.

```
#include <stdio.h>
#include  "units.h"
int main(void)
{
   float inches, centimetros;
   printf ( "DYNAMIC Escreva um valor polegadas:" );
   scanf("%f", &inches);
   .......
}
```

- Geração do executável convertsh.exe **$ cc –o convertsh.exe convertsh.c libunits.so**

Exemplo de execução dum programa que utilize bibliotecas dinâmicas

Problemas na execução do **convertsh.exe** ? Investigue a variável do shell LD_LIBRARY_PATH

$ LD_LIBRARY_PATH=$LD_LIBRARY_PATH:.
$ Export LD_LIBRARY_PATH