Construção de bibliotecas

Video desta aula

Bibliotecas são amplamente utilizadas na linguagem C. Além da biblioteca padrão (LibC) e das bibliotecas disponibilizadas pelo sistema operacional, o programador pode desenvolver suas próprias bibliotecas, para usar em seus projetos ou disponibilizá-las a terceiros.

As bibliotecas podem ser construídas para ligação estática ou dinâmica (DLL) com o código executável que as utiliza. Este texto explica as duas técnicas em ambiente Linux.

Estrutura geral

Vamos usar como exemplo uma biblioteca simples chamada *Hello*, que oferece funções para escrever na tela mensagens de "olá" em diversas linguagens. O código-fonte dessa biblioteca é composto pelos arquivos abaixo.

O arquivo de cabeçalho define a interface da biblioteca *Hello*:

hello.h

```
#ifndef __HELLO__
#define __HELLO__

void hello_pt () ;
void hello_en () ;
void hello_fr () ;

#endif
```

Os demais arquivos definem a implementação dessas funções:

hello_pt.c

```
#include <stdio.h>
#include "hello.h"

void hello_pt ()
{
  printf ("Ola, mundo!\n") ;
}
```

hello_en.c

```
#include <stdio.h>
#include "hello.h"
```

```
void hello_en ()
{
  printf ("Hello, world!\n") ;
}
```

hello_fr.c

```
#include <stdio.h>
#include "hello.h"

void hello_fr ()
{
  printf ("Salut, le monde !\n") ;
}
```

Um programa que utilize a biblioteca *Hello* pode ser escrito desta forma:

main.c

```
#include "hello.h"

int main ()
{
  hello_pt () ;
  hello_en () ;
  hello_fr () ;

  return 0 ;
}
```

Bibliotecas estáticas

Bibliotecas estáticas são ligadas ao programa durante o processo de compilação, resultando em um executável maior, mas menos dependentes das bibliotecas instaladas no sistema. Para construir uma biblioteca de ligação estática são necessários vários passos, descritos a seguir.

1) Inicialmente, todos os arquivos-fonte que irão compor a biblioteca devem ser compilados, para gerar seus arquivos-objeto correspondentes:

```
$ gcc -Wall -c hello_pt.c
$ gcc -Wall -c hello_en.c
$ gcc -Wall -c hello_fr.c
```

2) A seguir, deve ser usado o utilitário `ar` (*archiver*) para juntar todos os arquivos-objeto em uma biblioteca estática chamada `libhello.a`:

```
$ ar rvs libhello.a hello_pt.o hello_en.o hello_fr.o
```

Os flags `rvs` indicam:

- r (*replace*): substituir versões anteriores dos arquivos na biblioteca, caso existam
- v (*verbose*): mostrar na tela as inclusões que estão sendo realizadas
- s (*symbols*): criar uma tabela dos símbolos[1)] que estão sendo agregados à biblioteca

O utilitário `ar` possui diversos outros *flags*. Por exemplo, pode-se consultar o conteúdo de uma biblioteca estática:

```
$ ar t libhello.a
hello_en.o
hello_fr.o
hello_pt.o
```

Pode-se consultar todos os símbolos definidos em uma biblioteca estática (ou em qualquer arquivo objeto) através do utilitário `nm`:

```
$ nm libhello.a

hello-en.o:
0000000000000000 T hello_en
                 U puts

hello-fr.o:
0000000000000000 T hello_fr
                 U puts

hello-pt.o:
0000000000000000 T hello_pt
                 U puts
```

Para atualizar/incluir qualquer arquivo da biblioteca, basta executar `ar` novamente, indicando o(s) arquivo(s) a atualizar/incluir:

```
$ ar rvs libhello.a hello_it.o hello_es.o hello_jp.o
```

3) A forma mais simples de usar a biblioteca é indicá-la ao compilador no momento da compilação ou ligação:

```
$ gcc -Wall main.c -o main libhello.a
```

Uma opção abreviada de ligação pode ser utilizada. Nela, não é necessário indicar o nome completo da biblioteca:

```
$ gcc -Wall main.c -o main -L. -lhello
```

Esta abordagem é melhor que a anterior, pois neste caso o ligador somente irá incluir no executável final os objetos que forem efetivamente necessários.

A opção `-L.` é necessária para incluir o diretório corrente nos caminhos de busca de bibliotecas do ligador.

Observe que a biblioteca foi informada ao ligador na opção `-lhello`. Por default, ao encontrar uma opção `-labc`, o ligador irá procurar pela biblioteca `libabc.a` nos diretórios default de bibliotecas (`/lib`, `/usr/lib`, `/usr/local/lib`, ...) e depois disso nos diretórios informados pela opção `-L`.

Bibliotecas dinâmicas

Bibliotecas dinâmicas (DLLs) são ligadas ao programa durante a carga do executável na memória, resultando em um executável menores, mas que dependem das bibliotecas necessárias estarem instaladas no sistema operacional.

A construção de uma biblioteca de ligação dinâmica é um pouco mais complexa:

1) Primeiro, é necessário compilar os arquivos-fonte que irão compor a biblioteca usando a opção `-fPIC`, que irá gerar código binário independente de posição (PIC - Position Independent Code)[2]:

```
$ gcc -Wall -fPIC -c hello_pt.c
$ gcc -Wall -fPIC -c hello_en.c
$ gcc -Wall -fPIC -c hello_fr.c
```

2) A seguir, pode-se criar a biblioteca dinâmica, a partir dos arquivos-objeto:

```
$ gcc -Wall -g -shared -Wl,-soname,libhello.so.0 -o libhello.so.0.0 hello_pt.o
hello_en.o hello_fr.o
```

Observe que a opção `-Wl` transfere a opção `-soname=libhello.so.0` ao ligador. Essa opção permite definir o nome e versão da biblioteca.

3) Finalmente, para instalar a biblioteca, deve-se movê-la para o diretório adequado (geralmente `/usr/lib` ou `/usr/local/lib`)[3] e gerar os atalhos necessários para indicar os números de versão (0) e revisão (0):

```
# mv libhello.so.0.0 /usr/local/lib
# cd /usr/local/lib
# ln -s libhello.so.0.0 libhello.so.0
# ln -s libhello.so.0 libhello.so

$ ls -l
lrwxrwxrwx  1  prof      12   Out 2 18:20  libhello.so -> libhello.so.0
lrwxrwxrwx  1  prof      14   Out 2 18:06  libhello.so.0 -> libhello.so.0.0
-rwxr-xr-x  1  prof    6914   Out 2 18:06  libhello.so.0.0
```

4) A compilação usando a biblioteca ocorre da mesma forma que no caso estático:

```
$ gcc -Wall main.c -o main -L. -lhello
```

Ao carregar o executável, o sistema operacional irá localizar as bibliotecas dinâmicas necessárias, carregá-las e mapeá-las na área de memória do novo processo:

```
$ ./main
```

Caso a biblioteca esteja em um diretório não listado em `/etc/ld.so.conf` (arquivo de configuração do carregador e ligador dinâmico), ocorrerá um erro. Nesse caso, deve-se incluir o diretório nesse arquivo e a seguir executar `ldconfig`, ou informar o carregador dinâmico do SO através da variável de ambiente `LD_LIBRARY_PATH`:

```
$ export LD_LIBRARY_PATH=.
$ ./main
```

[1]
nomes de funções e de variáveis globais

2)

Como a ligação da biblioteca ocorre durante a carga/execução, a posição de seu código na memória dos processos que irão utilizá-la não pode ser determinada previamente.

3)

se for um diretório público, isso deve ser feito pelo administrador.