Programação de Computadores

Aula #13

Construindo Bibliotecas em C

O que são bibliotecas estáticas e dinâmicas? Como construir uma biblioteca em C? Como compilar um código com uma biblioteca em C?

Ciência da Computação - BCC e IBM - 2024/01
Prof. Vinícius Fülber Garcia

Bibliotecas Estáticas e Dinâmicas

A diferença de uma biblioteca estática e de uma biblioteca dinâmica está no momento da ligação destas com o programa que as utiliza!

Estática: a biblioteca é ligada ao programa em tempo de compilação. Ou seja, o código da biblioteca é incorporado ao executável do mesmo.

Dinâmica: a biblioteca é carregada e ligada ao programa em tempo de execução. Assim, a biblioteca deve ser previamente disponibilizada no sistema para ser utilizada pelo programa.

Bibliotecas Estáticas e Dinâmicas

Apesar de apresentarem características diferentes quanto à ligação em um programa, bibliotecas, sejam estáticas ou dinâmicas, apresentam uma estrutura comum.

Por exemplo, uma biblioteca é composta por um ou mais módulos!

Um módulo nada mais é do que arquivos contendo funções, definições, estruturas, etc... Tipicamente um módulo agrupa dados relacionados a um determinado escopo.

Bibliotecas Estáticas e Dinâmicas

Vamos analisar a criação de uma biblioteca (primeiro estática, depois dinâmica) que provê funções que executam o **programa "Olá Mundo" em várias línguas**.

Vamos organizar a biblioteca criando, inicialmente, **três arquivos de código (.c)**, o primeiro implementando a função em português, o segundo em inglês e o terceiro em francês.

Vejamos o resultado a seguir...

Bibliotecas Estáticas e Dinâmicas

hello_pt.c

```c
#include <stdio.h>

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

hello_en.c

```c
#include <stdio.h>

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

hello_fr.c

```c
#include <stdio.h>

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

Bibliotecas Estáticas e Dinâmicas

```
#ifndef __HELLO__
#define __HELLO__

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

#endif
```

Por fim, temos o arquivo de cabeçalhos (.h) que inclui as funções que compõem a biblioteca.

hello.h

Note que, não necessariamente as funções precisam estar no mesmo módulo de código (arquivo .c), podem ser vários.

Ligação Estática

Agora, que já temos os códigos-fonte da biblioteca preparados, devemos gerar o código objeto dos módulos da mesma (no caso, dos arquivos de código – .c):

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

Nesse ponto, teremos os arquivos objeto para todos os módulos da biblioteca!

Ligação Estática

O próximo passo é utilizar o **utilitário chamado *archiver* (ar)** para juntar todos os arquivos-objeto em uma **biblioteca chamada libhello.a**!

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

Onde as *flags* significam:

- r (replace): substitui versões anteriores dos arquivos na biblioteca
- v (verbose): mostra na tela as inclusões que estão sendo realizadas
- s (symbols): cria uma tabela dos símbolos agregados à biblioteca

Ligação Estática

Para consultar os módulos de uma biblioteca, podemos executar:

```
ar t libhello.a
```

Para verificar a tabela de símbolos de uma biblioteca, podemos usar o utilitário nm:

```
nm libhello.a
```

Ligação Estática

```c
#include "hello.h"

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

  return 0 ;
}
```

Desse ponto em diante, para utilizarmos a biblioteca *libhello* basta incluir o seu .h e indicar a mesma no processo de compilação de um programa.

gcc main.c -o main libhello.a

Os arquivos .a quanto o .h devem estar disponíveis!

Ligação Estática

Também, é possível informar a biblioteca no processo de compilação da seguinte forma:

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

A ***flag* -L indica os diretórios** onde as bibliotecas devem ser buscadas (além dos já tradicionais /lib, /usr/lib, /usr/local/lib, ...).

Já a ***flag* -l**, imediatamente seguida do nome da biblioteca (sem espaço) **serve como contração do nome** do seu arquivo (libhello.a).

Ligação Dinâmica

A construção de uma biblioteca dinâmica é um pouco mais complexa que a de uma estática!

Primeiro, é necessário compilar os arquivos fonte que irão compor a biblioteca usando a ***flag* -fPIC**, que irá gerar código binário independente de posição:

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

Códigos independentes de posição podem ser carregados em qualquer lugar da memória e ainda assim funcionarem corretamente.

Ligação Dinâmica

Em seguida, cria-se a biblioteca a partir da utilização do próprio compilador, transferindo ao ligador o nome e a versão da biblioteca através da ***flag* -Wl**:

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

O arquivo (DLL) da biblioteca *libhello.so.0.0* será então gerado e utilizado nos processos seguintes.

Ligação Dinâmica

Finalmente, para instalar a biblioteca, devemos **movê-la para o diretório adequado (geralmente /usr/lib ou /usr/local/lib) 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
```

Ligação Dinâmica

Caso a biblioteca esteja em um **diretório não listado em /etc/ld.so.conf**, ocorrerá um erro.

Assim, 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=caminho
sudo echo "caminho" > /etc/ld.so.conf.d/local.conf
```

Note que o .h das bibliotecas dinâmicas são tipicamente movidos para a pasta /usr/include

Ligação Dinâmica

Finalmente, **as configurações do ligador não são atualizadas** simplesmente adicionando o caminho da biblioteca à variável LD_LIBRARY_PATH, nem mesmo pelo fato de incluir o arquivo no diretório correto.

Para atualizar a lista de bibliotecas dinâmicas reconhecidas, é necessário executar o seguinte comando:

```
sudo ldconfig
```

Ligação Dinâmica

E a compilação do nosso programa que usa a biblioteca?

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

Outra opção é criar um arquivo .pc e prove-lo para uso pelo utilitário *pkg-config* (movendo-o para a pasta **/usr/lib/pkgconfig** ou **/usr/local/lib/pkgconfig**). Assim podemos fazer a compilação como mostrado na última aula.

Exercício #13

Crie uma **biblioteca dinâmica em C que contenha uma função chamada "soma"**, que recebe dois números inteiros como parâmetros e retorna a soma desses dois números. Em seguida, crie um programa principal que utilize essa biblioteca dinâmica para **calcular a soma de dois números** fornecidos pelo usuário.

Note que não poderemos instalar a biblioteca dinâmica nas pastas consideradas padrão no laboratório. Assim, mantenha a mesma na pasta local do seu programa principal e informe o diretório para realizar o processo de compilação.

Obrigado!

Vinícius Fülber Garcia
inf.ufpr.br/vinicius
vinicius@inf.ufpr.br