**Bibliotecas Compartilhadas**  
Entenda como diversos programas podem compartilhar bibliotecas comuns economizando recursos  
  
*Ricardo Caratti*  
[caratti@inep.gov.br](mailto:caratti@inep.gov.br)

Em programação, entende-se por bibliotecas, arquivos que contêm um conjunto de módulos ou membros de códigos pré-compilados reutilizáveis. Esses códigos podem ser usados por vários programas sem a necessidade de detalhes de sua implementação. A grande vantagem de usar uma biblioteca, é que uma vez fabricada, não será mais preciso compilar, bastando simplesmente ligá-la ao programa desejado. Dessa forma, existe uma grande vantagem em usar bibliotecas, pois uma vez implementada ou adquirida de terceiros, o desenvolvedor pode se abstrair dos detalhes e concentrar-se somente no problema principal. A tabela da página ao lado ilustra o uso de uma biblioteca compartilhada.

Note que "Meu Programa" necessitou usar uma função X e um procedimento Y. Entende-se por função, um procedimento ou uma rotina que retorna um valor. Sabendo que essas rotinas existem na biblioteca, para usá-las, basta fazer referência à biblioteca durante a compilação do programa principal.

Em Linux, pode-se desenvolver dois tipos de bibliotecas. A biblioteca de ligação estática e a biblioteca de ligação dinâmica. As bibliotecas estáticas são ligadas ao programa e fazem parte do arquivo executável. Já as bibliotecas dinâmicas são ligadas em tempo de execução, ou seja, a ligação ocorre por demanda. Portanto, não fazem parte do programa principal, reduzindo assim o tamanho do arquivo executável.

Optar pelo uso de bibliotecas estáticas ou dinâmicas depende muito do que se pretende. Em geral, pode-se obter o mesmo resultado usando uma ou outra técnica. Bibliotecas estáticas deixam o código executável mais livre da configuração do ambiente, ou seja, todo o código que o programa precisa para ser executado já se encontra no próprio executável. Ao contrário, quando se faz uso de bibliotecas compartilhadas, o programa é dividido em um módulo principal e em um ou mais módulos que serão ligados dinamicamente. Considerando que bibliotecas compartilhadas podem ser usadas por mais de um programa ao mesmo tempo, elas ocupam menos memória RAM, menos espaço em disco, menos recursos do sistema, é mais simples fazer manutenção.

Resumindo, bibliotecas são arquivos que contêm módulos reutilizáveis pré-compilados que serão usados por desenvolvedores de aplicações. Elas podem ser classificadas em estáticas e compartilhadas. Ao optar por bibliotecas estáticas, ela passará a fazer parte do corpo do programa principal, liberando-o assim do ambiente de configuração do sistema. Ao contrário, bibliotecas compartilhadas ligam-se ao programa principal dinamicamente, ou seja, um módulo só será ligado ao programa principal se for solicitado. Com isso, bibliotecas compartilhadas consomem menos recursos do sistema operacional além de facilitarem a substituição de módulos defeituosos sem a necessidade de compilar novamente todos os outros módulos ou sistemas envolvidos.

Para ilustrar o desenvolvimento de bibliotecas compartilhadas, será utilizado o ambiente de desenvolvimento na linguagem C que vem com a própria distribuição Linux.

Será criado um procedimento que mostrará o resultado da soma de dois valores e uma função que dirá qual o maior valor.

Note pelo fonte a seguir, que não existe nada incomum no desenvolvimento das rotinas.

Fonte: MinhaLib.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

/\* Dados dois argumentos inteiros, a e b, imprime a soma. \*/

void MinhaProcSoma( int a, int b)

{

int c;

c = a + b;

printf("\n a + b = %d \n", c);

}

/\* Dado dois argumentos inteiros, a e b, retorna o maior valor \*/

int MinhaFuncMaiorValor( int a, int b)

{

if ( a >= b)

return a;

else

return b;

}

É preciso agora tornar a função e o procedimento disponíveis para qualquer programa. Para tanto, é preciso compilá-lo utilizando a seguinte linha de comando:

#gcc -fPIC -c MinhaLib.c

Para obter mais detalhes sobre as opções de compilação do C da GNU utilize o comando man gcc. Um manual completo das opções do compilador será mostrado.

Após a compilação, um arquivo MinhaLib.o será gerado.

Até agora, tudo que se tem é um módulo compilado com um procedimento e uma função. Nessa forma, é possível utilizar MinhaLib.o como uma biblioteca estática. Bastando para isso, ligá-la ao programa principal da seguinte forma:

#gcc -o ProgPrincipal ProgPrincipal.c MinhaLib.o

Para fazer uso dos benefícios de bibliotecas compartilhadas, é preciso ir um pouco mais além. Para tanto, é preciso usar a seguinte linha de comando:

#gcc -shared -Wl,-soname,libMinhaLib.so.1 -o libMinhaLib.so.1.0

MinhaLib.o

Apesar de não ser obrigatório iniciar o nome de uma biblioteca compartilhada com lib, é muito desejável, já que é o padrão de algumas ferramentas do Linux como veremos mais adiante. Pelo comando acima, será criada uma biblioteca com o nome de libMinhaLib.so.1.0. O sistema operacional irá reconhecê-la pelo nome de libMinhaLib.so.1. Isso também não é obrigatório, mas é bom usar esse padrão para ficar de acordo com as normas de desenvolvimento do Linux.

Na realidade, o Linux usa o seguinte padrão para bibliotecas compartilhadas:

libaaa.so.b.c.ddd

* libaaa.soé o nome da biblioteca;
* b é um número que indicará a maior versão;
* c é um número que indicará a menor versão;
* ddd é um número que informa a "release".

Uma vez desenvolvidas as rotinas e criada a biblioteca compartilhada,

é preciso publicá-la. Para tanto será preciso

escolher um diretório onde suas bibliotecas ficarão. De

preferência utilize /usr/local/lib. Utilizando um editor de texto de sua

preferência, altere o arquivo /etc/ld.so.conf adicionando na última linha o

diretório /usr/local/lib.

Mova a biblioteca compartilhada libMinhaLib.so.1.o para /usr/local/lib

#mv libMinhaLib.so.1.o /usr/local/lib

em seguida execute o comando:

# /sbin/ldconfig

o utilitário ldconf irá a todos os diretórios existentes

referenciados em ld.so.conf e carregará todas as bibliotecas

compartilhadas que iniciarem com lib, criando também um link que de acordo com

o exemplo será libMinhaLib.so.1. Também será

criada uma referência para cada biblioteca compartilhada em /etc/ld.so.cache.

Isso permitirá que ela seja carregada toda vez que o sistema iniciar.

Após executar ldconfig, verifique no diretório /usr/local/lib

os sequintes arquivos:

* libMinhaLib.so.1(Link criado pelo ldconfig);
* libMinhaLib.so.1.o a biblioteca criada por você.

Veja o teste do uso da biblioteca na figura 1, abaixo.

|  |
| --- |
| **Figura 1**  Programa de Teste (TestaProc.c):  #include <dlfcn.h>  #include <stdio.h>  int main()  {  void \*descritor;  void (\*PonteiroParaMinhaProcSoma)( int a, int b);  int (\*PonteiroParaMinhaFuncMaiorValor( int a, int b);  int x, y, MaiorValor;  x = 10; /\* Somente para exemplo \*/  y = 11; /\* Somente para exemplo \*/  /\* Abre a biblioteca compartilhada \*/  if ( descritor = dlopen("MinhaLib.so.1.0",RTLD\_LAZY))  {  PonteiroParaMinhaProcSoma = dlsym(descritor,"MinhaProcSomal");  PonteiroParaMinhaFuncMaiorValor =  dlsym(descritor,"MinhaFuncMaiorValor");  /\* Chamada a MinhaProcSoma \*/  (\*PonteiroParaMinhaProcSoma)( 2, 4 );  (\*PonteiroParaMinhaProcSoma)( x, y );  /\* Chamada a MinhaFuncMaiorValor \*/  MaiorValor = (\*PonteiroParaMinhaFuncMaiorValor)(x,y);  printf("\nO Maior valor é: %d", MaiorValor);  }  else  {  printf("\nErro ao tentar usar a biblioteca dinâmica\n");  printf("\n%s\n",dlerror() );  return 0;  }  } |

Compile esse programa com:

# gcc -o TesteProc TesteProc.c -ldl

Para executar faça:

# ./TestaProc

O exemplo mostrará o seguinte resultado:

a + b = 6

a + b = 21

O maior valor é: 11

Para efeito prático, altere o procedimento MinhaProcSoma em

MinhaLib.c da seguinte forma:

c = b - a;

printf("\n b - a = %d \n", c);

Execute os passos a seguir:

# gcc -fPIC -c MinhaLib.c

# gcc -shared -Wl,-soname,libMinhaLib.so.1 -o libMinhaLib.so.1.0 MinhaLib.o

# mv libMinhaLib.so.1.o /usr/local/lib

Execute novamente o programa TestaProc da seguinte forma:

# ./TestaProc

Resultado:

b - a = 2

b - a = 1

O maior valor é: 11

Note que não foi preciso compilar novamente o programa TestaProc.

Nesse caso, a ligação à biblioteca alterada MinhaLib,

foi efetuada em tempo de execução.

Concluindo, o Linux bem como outros ambientes Unix facilitam o uso

e o desenvolvimento de bibliotecas compartilhadas. Dentre as vantagens

em usar essas bibliotecas, destaca-se a facilidade de manutenção

de sistemas, considerando, é claro, que para tanto basta trocar

o módulo defeituoso sem a necessidade de re-compilar todos os

outros módulos que o compõem.

|  |
| --- |
| =============================================================  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  | | | | | |  | | Meu Programa | | Biblioteca | |  | | (Programa Principal) | | (Coleção de módulos) | |  | |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_| |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_| |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  | | | | | |  | | Chamada a Função X() | <=> | Função X() | |  | | A = X() | |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_| |  | |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_| |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  | | | | | |  | | Chamada ao Procedimento Y()| <=> | Procedimento Y() | |  | |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_| |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_| |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  | | | | | |  | | | | Procedimento Y() | |  | |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_| |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_| |  | |  ============================================================= |

|  |
| --- |
| Tabela de observações  **#include <dlfcn.h>** Declarações das funções para manipulação de bibliotecas dinâmicas  **dlopen (const char\*filename, int flag)** Carrega uma biblioteca Compartilhada  **void \* dlsym(void \*handle, char \*symbol)** Obtém o endereço do procedimento dentro da biblioteca compartilhada  **const char \*dlerror()** Retorna uma string contendo o erro ou NULL se tudo OK  **int dlclose(void \*handle)** Fecha a biblioteca |

|  |
| --- |
| **para saber mais** |
| * Wall, Watson, and Whitis. Linux Programming. Sams, 1999 * Barkakati, Naba. Red Hat Linux Secrets, Info World - IDG Books, 2ª Edição, 1999 |