**Criação de Bibliotecas Dinâmicas em C**

Se você já tentou instalar aplicações em sistemas Linux ou em Unix-likes sem a ajuda de um gerenciador de pacotes provavelmente, em algum momento já se deparou com erros como esse:

error while loading shared libraries: libpthread\_rt.so: cannot open shared object file: No such file or directory

Neste posts eu quero ensinar como criar esses erros :). Quero explicar como criar bibliotecas que são linkadas em tempo de execução, porém primeiro quero dar uma pequena introdução no processo de compilação e mostrar vantagens e desvantagens desse modo de criação de bibliotecas.

**O Processo da Compilação**

O que significa compilar um código?

Compilação é o processo que traduz uma determinada linguagem em outra. Neste artigo focaremos o processo de compiação das linguagens C e C++, neste caso o processo traduz o código escrito utilizando a gramatica da linguagem que é compreensível pelos seres humanos(pelo menos por alguns) para um código que o sistema operacional e hardware entendam, chamado código objeto. Este código ainda não é nosso programa final, ele ainda não pode ser executado. Após a compilação deve acontecer um processo chamado linkagem. Onde códigos são acoplados no código do nosso programa. Por exemplo, quando criamos um simples "Hello World!" em C, não estamos criando a função printf, mas ela deve estar no código para o programa ser executado, por isso o #include<stdio.h> é necessário. Após a compilação do código as instruções da função printf ainda não está presente, é no processo de linkagem que código do printf(e muitos outros) é incorporado ao nosso.

/\*

file: hello.c

\*/

#include <stdio.h>

int main(){

printf("Hello World!\n");

return 0;

}

#compilando o codigo

gcc hello.c -c

#linkagem

gcc -o hello hello.o

Quando criamos várias bibliotecas separadas, como o código abaixo, todo este código será incorporado no arquivo final de nosso programa.

/\*

arquivo: fatorial.c

\*/

#include "fatorial.h"

unsigned long int fatorial(unsigned long int valor)

{

unsigned long total = 1;

int i;

for(i=valor;i>=2;i--)

total \*= i;

return total;

}

/\*

arquivo: fatorial.h

\*/

unsigned long int fatorial(unsigned long int valor);

/\*

arquivo: program.c

\*/

#include <stdio.h>

#include "fatorial.h"

int main(){

unsigned long int valor;

printf("Digite um valor:\n");

scanf("%lu", &amp;valor);

printf("O fatorial de %lu é %lu\n",valor,fatorial(valor));

return 0;

}

O programa acima pode ser gerado com os seguintes comandos:

gcc -c fatorial.c

gcc -o program program.c fatorial.o

Quando criamos o arquivo final do programa, todo o código do fatorial está embutido *fisicamente<* nele. Se for necessário alterar algo em fatorial.c o programa precisa ser recompilado e é ai que problemas podem acontecer. "Como assim? É só executar esses dois comandos e pronto", neste caso sim, pois o exemplo é bem simples, mas imagine por exemplo se alguma falha é encontrada em uma biblioteca um pouco mais usada como libssl(só por curiosidade fica o link ai: [Comunicado do Projeto Debian sobre falha no SSL](http://lists.debian.org/debian-security-announce/2008/msg00152.html), quantas aplicações precisariam ser recompiladas? Ou por exemplo no Qt ou Gtk, quantas aplicações usam essas libs? Outro problema, se tivermos varias aplicações usando as mesmas rotinas e essas rotinas sempre ficam dentro do binário final, ocupando espaço em disco e principalmente, em memória, um grande desperdício é gerado. Neste momento que entram as Bibliotecas Dinâmicas(compartilhadas). Essas bibliotecas são linkadas ao programa em tempo de execução e as rotinas não estão no binário do programa.

**Utilizando Bibliotecas Dinâmicas**

No mundo Unix-like e Linux as bibliotecas dinâmicas possuem o seguinte padrão de nome: *lib<nome>.so.<versão>*, sendo que a versão é opcional e o <nome> será passado para o compilador e por fim *so* significa *shared object*. As bibliotecas devem estar em diretórios específicos como os */lib* e */usr/lib*, porém novos diretórios podem ser adicionados, basta adiciona-los na variável de ambiente LD*LIBRARY*PATH(cada diretório dever estar separados por ':', da mesma forma que a variável PATH) ou utilizando o comando ldconfig, este pode receber como parâmetro o diretório com caminho absoluto onde estão as libs.

Para compilar o programa utilizando uma biblioteca dinâmica o parâmetro *-l* deve ser passado seguido do nome da lib, como por exemplo *-lm* ou *-lglut*. Para vermos quais libs serão requiridas por um programa podemos utilizar o comando *ldd <comamdo>*.

A outra forma de utilização do biblioteca dinâmica, carrega-la em tempo de execução, dentro do código.

**Criando Bibliotecas Dinâmicas**

Não precisamos alterar os arquivo fatorial.c e fatorial.h para criar nossa biblioteca dinâmica, precisamos sim alterar a forma de compilação dos mesmos. Na compilação do fatorial.c passaremos agora o parâmetro *-shared* para o gcc, a opção *-fPIC* e a opção *-o* com valor *libfatorial.so*.

gcc -shared -fPIC fatorial.c -o libfatorial.so

#para criação do programa utilizando a lib

gcc -o program program.c -L. -lfatorial

A resposta do programa será a mesma com a compilação anterior. Um teste legal que pode ser feito neste momento é alterar o valor retornado na função fatorial para um qualquer, mesmo que errado e recompile apenas o lib e execute o programa, você ira notar que o programa alterou o resultado sem ser compilado novamente.

Essa forma de uso não irá embutir o código da lib durante a linkagem mas será linkado durante a execução do programa. Se a lib não for encontrada nos diretórios configurados o programa abortara apontando o erro descrito no começo do post.

Há uma outra forma de uso da biblioteca dinâmica, nós podemos carregar a lib durante a execução no nosso programa usando a lib dl. Abaixo está um programa carregando a lib fatorial em tempo de execução.

/\*

file: program\_load.c

\*/

#include<stdio.h>

#include<stdlib.h>

#include<dlfcn.h>

int main()

{

void \*handle;

unsigned long int (\*fatorial)(unsigned long int);

unsigned long int valor;

char \*error;

handle = dlopen("libfatorial.so.1.0",RTLD\_LAZY);

if(!handle)

{

fprintf(stderr, "%s\n", dlerror());

exit(1);

}

fatorial = dlsym(handle, "fatorial");

if((error = dlerror()) != NULL)

{

fprintf(stderr,"%s\n", error);

exit(1);

}

printf("Digite um valor:\n");

scanf("%lu", &amp;valor);

printf("O fatorial de %lu é %lu\n",valor,(\*fatorial)(valor));

dlclose(handle);

return 0;

}

O código acima pode ser compilado com o seguinte comando:

gcc -o program\_load program\_load.c -ldl

Se a lib não for encontrada o programa poderá ser iniciado, diferente da forma de linkagem anterior, assim podemos tratar a ausência da biblioteca. Esta forma é muito útil na criação de plugins para programas.

Mais informações sobre as funções da lib dl pode ser encontrados no man pages, por exemplo: *man dlopen*.