Gerenciar bibliotecas

Sumário

Índice de tabelas

Índice de Figuras

Capítulo 1

Gerenciar bibliotecas

- Como identificar bibliotecas estáticas e dinâmicas.
- Balanço entre tipos de bibliotecas;
- Funcionamento (comandos, arquivos de configuração e diretórios).

1.1. Mãos a obra

A bibliotecas atribuem diversas funcionalidades a um binário quando executado, essas funções permitem que os binários possam exibir informações na tela, escrever dados em um disco e/ou manipular recurso em um rede.

Mas como identificar que tipo de biblioteca um binário esta utilizando?

Um comando muito útil é o ldd, onde exibe quais bibliotecas compartilhadas são necessárias para o binário em tempo de execução.

```
# ldd $(which cp)
```

```
        linux-gate.so.1 =>  (0xb7f9e000)
        libselinux.so.1 => /lib/libselinux.so.1 (0xb7f7a000)
        libacl.so.1 => /lib/libacl.so.1 (0xb7f73000)
        libc.so.6 => /lib/i686/cmov/libc.so.6 (0xb7e17000)
        libdl.so.2 => /lib/i686/cmov/libdl.so.2 (0xb7e13000)
        /lib/ld-linux.so.2 (0xb7f9f000)
        libattr.so.1 => /lib/libattr.so.1 (0xb7e0e000)
```

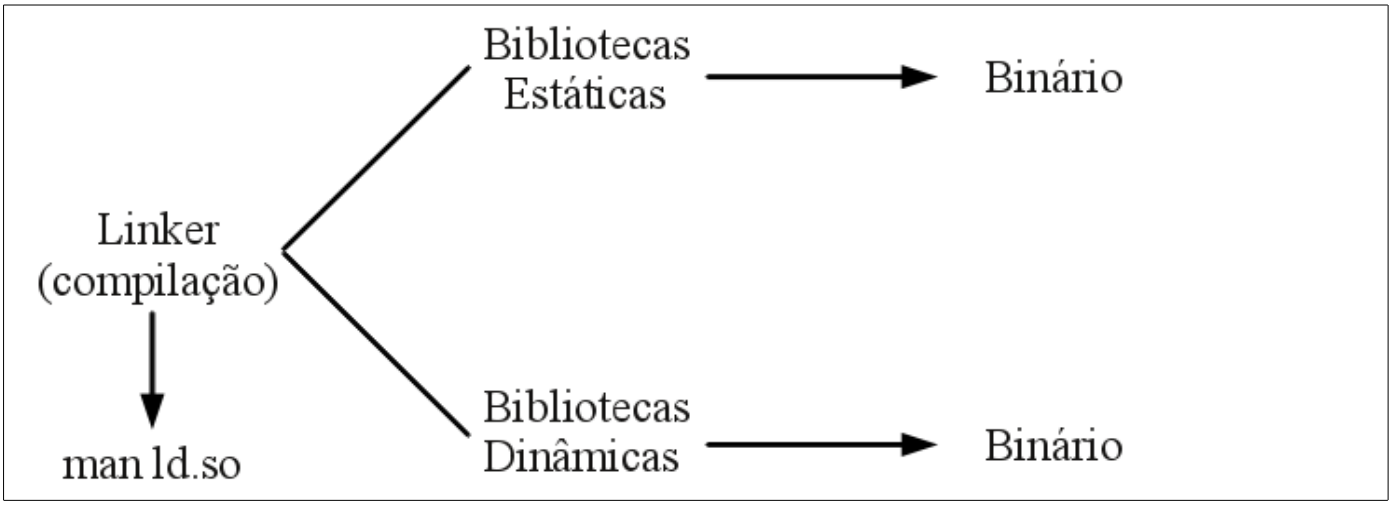
Em nosso exemplo foram listadas todas as bibliotecas que o comando cp utiliza. Use o mesmo comando agora para o binário zcat.

```
# ldd $(which zcat)
```

```
        not a dynamic executable
```

Já o binário zcat não usa bibliotecas compartilhadas.

A diferença é que um binário com bibliotecas estáticas, as contem em seu interior, já o binário com bibliotecas compartilhadas possui apenas uma referencia.



Para visualizar todas as bibliotecas de todos os binários de um diretório, use o comando ls em conjunto com xargs, which e ldd. Assim é possível descobrir de uma só vez, quais binários utilizam bibliotecas compartilhadas e estáticas.

```
# ls /bin | xargs which | xargs ldd
```

```
/bin/uncompress:
        not a dynamic executable
/bin/vdir:
        linux-gate.so.1 =>  (0xb7fbd000)
        librt.so.1 => /lib/i686/cmov/librt.so.1 (0xb7fa9000)
        libselinux.so.1 => /lib/libselinux.so.1 (0xb7f90000)
        libacl.so.1 => /lib/libacl.so.1 (0xb7f88000)
        libc.so.6 => /lib/i686/cmov/libc.so.6 (0xb7e2d000)
        libpthread.so.0 => /lib/i686/cmov/libpthread.so.0 (0xb7e14000)
        /lib/ld-linux.so.2 (0xb7fbe000)
        libdl.so.2 => /lib/i686/cmov/libdl.so.2 (0xb7e10000)
        libattr.so.1 => /lib/libattr.so.1 (0xb7e0b000)
/usr/bin/which:
        not a dynamic executable
/bin/zcat:
        not a dynamic executable
/bin/zcmp:
        not a dynamic executable
```

Balanço entre tipos de bibliotecas

Fazendo um balanço entre os tipos de binários, temos mais performance quando usadas bibliotecas estáticas e mais gasto de memoria quando usadas as compartilhadas. Veja abaixo um comparativo com pontos positivos (+) contra os negativos (-).

Bibliotecas estáticas

Binário roda de forma independente **(+)**

Ganho de performance **(+)**

Gasto desnecessário de memória **(-)**

Tamanho maior no final do arquivo **(-)**

Bibliotecas dinâmicas (compartilhadas)

Gasto menor de memória **(+)**

Tamanho menor final do arquivo **(+)**

Linker faz referência em tempo de execução **(-)**

Perda de performance **(-)**

Funcionamento

Para entender todo o funcionamento das bibliotecas, vamos conhecer os comandos, arquivos, diretórios e suas ligações.

ldd

Retorna uma lista das bibliotecas dinâmicas que o binário precisa em tempo de execução. Muito útil para resolver problemas de falta de alguma biblioteca.

ld.so

Completa a ligação entre o binário e as bibliotecas dinâmicas em tempo de execução.

/etc/ld.so.cache

Arquivo de cache usado por ld.so para localizar as bibliotecas, deixando a ligação mais rapida.

/lib e **/usr/lib**

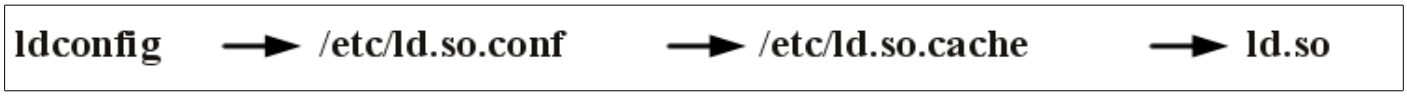
Localização das bibliotecas no sistema.

/etc/ld.so.conf

Arquivo com a localização de bibliotecas adicionais. É neste arquivo que temos informações de onde podemos personalizar caminhos adicionais para as bibliotecas

ldconfig

Comando usado para que as alterações no arquivo /etc/ld.so.conf atualizem o /etc/ld.so.cache. Exemplo:



Capítulo 2

Gerenciando

2.1. Objetivos

•Trobleshooting: Resolver problemas de bibliotecas.

2.1. Troubleshooting

Como posso resolver problemas de bibliotecas?

O comando ldconfig pode atualizar o cache resolvendo alguns problemas como falta de bibliotecas em novos programas instalados apenas descompactando um tar.gz. Como exemplo vamos utilizar o Kompozer, que um editor de páginas Web para Linux. Vamos a prática:

Primeiro baixe o pacote do Kompozer no endereço abaixo:

http://sourceforge.net/projects/kompozer/files/current/0.8b3/linux-i686/kompozer-0.8b3.ko.gcc4.2-i686.tar.gz/download

Descompacte para o diretório /usr/local através do comando tar

```
# tar xzvf kompozer-0.8b3.pt-BR.gcc4.2-i686.tar.gz -C /usr/local
```

Com um usuário comum execute o programa digitando seu caminho completo

```
$ /usr/local/kompozer/kompozer-bin
```

```
/usr/local/kompozer/kompozer-bin: error while loading shared libraries: libmozjs
.so: cannot open shared object file: No such file or directory
```

Veja em nosso exemplo que o ld.so não consegue achar as bibliotecas. Para resolver este problema, crie um arquivo de nome kompozer.conf no diretório /etc/ld.so.conf.d

```
# vim /etc/ld.so.conf.d/kompozer.conf
/usr/local/kompozer
```

No conteúdo do arquivo foi definido o caminho das bibliotecas necessárias para que o binário kompozer-bin funcione.

Agora para que o ld.so faça a ligação em tempo de execução das bibliotecas ao binário kompozer-bin, use o comando ldconfig.

```
# ldconfig
```

O que aconteceu?

O ldconfig leu a configuração do arquivo ld.so.conf que aponta para o diretório /etc/ld.so.conf.d, e neste diretório foi criado o arquivo kompozer.conf com a localização das bibliotecas do binário kompozer-bin.

Essa localização foi adicionada ao arquivo de cache ld.so.cache, que é consultado pelo ld.so para fazer as ligações das bibliotecas.

ldconfig → /etc/ld.so.conf → /etc/ld.so.cache → ld.so

Para testar o funcionamento, execute no modo gráfico através de um usuário comum, o nome do binário com se caminho completo. Também funciona usando as teclas ALT + F2 e digitando /usr/local/kompozer/kompozer-bin

```
$ /usr/local/kompozer/kompozer-bin
```