Bibliotecas no GNU/Linux

Introdução:

Hoje em dia é muito simples instalar um programa já compilado, com a ajuda de gerenciadores de pacotes como o pkgtools, rpm, dpkg, apt etc. Mas você vai encontrar muitos programas disponíveis somente em código-fonte, e às vezes nem tão bem documentados assim. Entretanto, compilar um programa não é algo de outro mundo, não é um bicho de sete cabeças .

Algo que devemos estar atentos são as bibliotecas. A função destas bibliotecas lembra um pouco a dos arquivos .dll no Windows. Temos as bibliotecas estáticas e dinâmicas. As dinâmicas são usadas por vários programas e necessárias para instalar programas distribuídos em código fonte (os famosos arquivos tar.gz)

Em sistemas Linux existem dois tipos fundamentais de programas executáveis. O primeiro é chamado de **estático**. Esse tipo de programa contém todas as funções que ele precisa para ser executado, em outras palavras, é completo. Devido a isso, os executáveis estáticos não dependem de nenhuma biblioteca externa para funcionar.

O segundo tipo é o executável **dinâmico**.

Mas como descobrir se um executável é dinâmico ou estático?

Para isso, podemos usar o comando abaixo, que produz uma lista de dependências:

 # ldd <caminho_do_executável>

Obs: Deve ser colocado o caminho completo do executável, não somente o nome do comando.

Para facilitar em vez de digitar o caminho completo:

 # ldd `which ls` (o comando which + comando desejado entre crases)

Exemplo:

```
root@gladiador# ldd /sbin/sln
        not a dynamic executable

root@gladiador# ldd /bin/ln
        linux-gate.so.1 =>  (0xffffe000)
        libc.so.6 => /lib/tls/libc.so.6 (0xb7ded000)
        /lib/ld-linux.so.2 (0xb7f29000)
```

Vamos ver agora as diferenças:

```
root@gladiador# du -h /bin/sln
428K    /bin/sln
root@gladiador# du -h /bin/ln
24K     /bin/ln
```

Note que um executável estático é bem maior que o executável dinâmico, isso ocorre pois o estático já contém o que precisa dentro do próprio executável. Obviamente bibliotecas compartilhadas tendem a gerar executáveis menores; eles também usam menos memória e significa que menos espaço em disco é usado.

O modo estático é ligeiramente mais rápido, pois não precisa buscar bibliotecas em diretórios, mas consome mais espaço (dado que cada programa teria uma cópia da biblioteca dentro de si).

O modo compartilhado é ligeiramente mais lento, pois precisa sempre abrir o arquivo da biblioteca, mas ocupa menos espaço (dado que só se tem uma cópia da biblioteca) e facilita centralizando a manutenção (se você precisar mudar a versão de uma biblioteca, não tem de recompilar o programa, basta trocar o arquivo da biblioteca). O padrão é usar bibliotecas compartilhadas, e geralmente é a decisão mais sábia, mas precisa que todas as bibliotecas necessárias estejam presentes no sistema para executar.

No Linux, **bibliotecas estáticas** têm nomes como **libname.a**, enquanto que bibliotecas **compartilhadas** são chamadas **libname.so.x.y.z** onde **x.y.z** é alguma forma de número de versão.

A última fase do desenvolvimento de um software é "linká-lo", ou seja, reunir todas as partes fundamentais para haver execução. Existem tarefas que a maioria dos softwares irá querer realizar como abrir arquivos, por exemplo, e esse tipo de tarefa é realizada através de bibliotecas. No Linux, as bibliotecas podem ser encontradas em **/lib** e **/usr/lib/** ou em outros diretórios.

Um exemplo real é a linguagem C que é rica em poder de expressão, mas relativamente pobre em funcionalidades. Para construir aplicações que fazem uso de funcionalidades específicas, como interfaces gráficas, comunicação via rede, fórmulas matemáticas complexas, etc, devem ser usadas **bibliotecas.**

As bibliotecas mais comuns, utilizadas por todas as aplicações e utilitários do sistema, são:

- **libc:** na verdade um grande "pacote" de bibliotecas que provê funcionalidades básicas de entrada/saída, de acesso a serviços do sistema, à rede, etc.
- **ld-linux:** provê as funções necessárias para a carga de bibliotecas dinâmicas, durante a inicialização do programa.

Por default, essas duas bibliotecas são **automaticamente** incluidas e ligadas em todos os programas.

Ao usar uma biblioteca estática, o linker encontra as partes que os módulos do programa precisam, e as copia fisicamente no arquivo de saída executável que ele gera. Para bibliotecas compartilhadas, não --- ao invés disso, ele deixa uma nota na saída dizendo "quando este programa for executado, ele terá de carregar primeiro esta biblioteca". Diversos programas, para não terem sempre de reinventar a roda, usam bibliotecas, como a libc, por exemplo.

Vamos simular uma situação real:

```
root@gladiador# ldd /usr/bin/xmms
        linux-gate.so.1 =>  (0xffffe000)
        libSM.so.6 => /usr/X11R6/lib/libSM.so.6 (0xb7f1c000)
        libICE.so.6 => /usr/X11R6/lib/libICE.so.6 (0xb7f04000)
        libXxf86vm.so.1 => /usr/X11R6/lib/libXxf86vm.so.1 (0xb7eff000)
        libxmms.so.1 => /usr/lib/libxmms.so.1 (0xb7ef2000)
        libgtk-1.2.so.0 => /usr/lib/libgtk-1.2.so.0 (0xb7dd4000)
        libgdk-1.2.so.0 => /usr/lib/libgdk-1.2.so.0 (0xb7da3000)
        libgmodule-1.2.so.0 => /usr/lib/libgmodule-1.2.so.0 (0xb7d9f000)
        libgthread-1.2.so.0 => /usr/lib/libgthread-1.2.so.0 (0xb7d9c000)
        libglib-1.2.so.0 => /usr/lib/libglib-1.2.so.0 (0xb7d7b000)
        libpthread.so.0 => /lib/tls/libpthread.so.0 (0xb7d69000)
        libdl.so.2 => /lib/tls/libdl.so.2 (0xb7d65000)
        libXext.so.6 => /usr/X11R6/lib/libXext.so.6 (0xb7d57000)
        libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0xb7c8c000)
        libm.so.6 => /lib/tls/libm.so.6 (0xb7c69000)
        libc.so.6 => /lib/tls/libc.so.6 (0xb7b4d000)
        /lib/ld-linux.so.2 (0xb7f45000)
```

Neste exemplo, todas as bibliotecas foram encontradas e carregadas e se encontram listadas ao lado direito. Para instalar uma biblioteca necessária ausente, devemos copiá-la em um dos diretórios listados no arquivo **/etc/ld.so.conf** e em seguida executar o comando **ldconfig**.

Opcionalmente, podemos editar o arquivo e colocar outros diretórios para o **ldconfig** procurar as bibliotecas, e depois reexecutá-lo. Para verificar todas as bibliotecas presentes em nossa máquina usamos **ldconfig -p**.

Agora, para vermos a importância disso vamos fazer uma simulação, vamos esconder uma biblioteca da aplicação xmms (player de música). Então, mudando o nome da biblioteca:

root@gladiador# **mv /usr/lib/libxmms.so.1 /usr/lib/teste**

Vamos tentar agora rodar a aplicação:

root@gladiador# **xmms**

xmms: error while loading shared libraries: **libxmms.so.1**: cannot open shared object file: No such file or directory

Ele não está conseguindo detectar **libxmms.so.1,** pois fizemos a simulação de alterar o nome dela.

Vamos checar as bibliotecas que faltam:

root@gladiador# **ldd /usr/bin/xmms**

```
linux-gate.so.1 =>  (0xffffe000)
libSM.so.6 => /usr/X11R6/lib/libSM.so.6 (0xb7f9d000)
libICE.so.6 => /usr/X11R6/lib/libICE.so.6 (0xb7f85000)
libXxf86vm.so.1 => /usr/X11R6/lib/libXxf86vm.so.1 (0xb7f80000)
libxmms.so.1 => not found
libgtk-1.2.so.0 => /usr/lib/libgtk-1.2.so.0 (0xb7e62000)
libgdk-1.2.so.0 => /usr/lib/libgdk-1.2.so.0 (0xb7e31000)
libgmodule-1.2.so.0 => /usr/lib/libgmodule-1.2.so.0 (0xb7e2d000)
libgthread-1.2.so.0 => /usr/lib/libgthread-1.2.so.0 (0xb7e2a000)
libglib-1.2.so.0 => /usr/lib/libglib-1.2.so.0 (0xb7e09000)
libpthread.so.0 => /lib/tls/libpthread.so.0 (0xb7df7000)
libdl.so.2 => /lib/tls/libdl.so.2 (0xb7df3000)
libXext.so.6 => /usr/X11R6/lib/libXext.so.6 (0xb7de5000)
libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0xb7d1a000)
libm.so.6 => /lib/tls/libm.so.6 (0xb7cf7000)
libc.so.6 => /lib/tls/libc.so.6 (0xb7bdb000)
/lib/ld-linux.so.2 (0xb7fc6000)
```

Vamos consertar isso:

root@gladiador# **mv /usr/lib/teste /usr/lib/libxmms.so.1**

Assim colocamos o nome original da biblioteca.

Vamos checar se a biblioteca está sendo detectada novamente:

root@gladiador# **ldd /usr/bin/xmms**
```
linux-gate.so.1 =>  (0xffffe000)
libSM.so.6 => /usr/X11R6/lib/libSM.so.6 (0xb7ef8000)
libICE.so.6 => /usr/X11R6/lib/libICE.so.6 (0xb7ee0000)
libXxf86vm.so.1 => /usr/X11R6/lib/libXxf86vm.so.1 (0xb7edb000)
libxmms.so.1 => /usr/lib/libxmms.so.1 (0xb7ece000)
libgtk-1.2.so.0 => /usr/lib/libgtk-1.2.so.0 (0xb7db0000)
libgdk-1.2.so.0 => /usr/lib/libgdk-1.2.so.0 (0xb7d7f000)
libgmodule-1.2.so.0 => /usr/lib/libgmodule-1.2.so.0 (0xb7d7b000)
libgthread-1.2.so.0 => /usr/lib/libgthread-1.2.so.0 (0xb7d78000)
libglib-1.2.so.0 => /usr/lib/libglib-1.2.so.0 (0xb7d57000)
libpthread.so.0 => /lib/tls/libpthread.so.0 (0xb7d45000)
libdl.so.2 => /lib/tls/libdl.so.2 (0xb7d41000)
libXext.so.6 => /usr/X11R6/lib/libXext.so.6 (0xb7d33000)
libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0xb7c68000)
libm.so.6 => /lib/tls/libm.so.6 (0xb7c45000)
libc.so.6 => /lib/tls/libc.so.6 (0xb7b29000)
/lib/ld-linux.so.2 (0xb7f21000)
```

O arquivo **/etc/ld.so.conf** contém os diretórios onde procurar dependências.

root@gladiador# **cat /etc/ld.so.conf**
```
/usr/local/lib
/usr/X11R6/lib
/usr/i486-slackware-linux/lib
/opt/kde/lib
/usr/lib/qt/lib
```

Obs:
/lib e /usr/lib já estão automaticamente inclusas!

Existe também a variável de ambiente **LD_LIBRARY_PATH**, que instrui o carregador dinâmico checar um certo diretório. Exemplo:
 # export LD_LIBRARY_PATH="/usr/lib/velho:/opt/lib"

Temos também outro arquivo, o **/etc/ld.so.cache** que contém uma lista de todas as bibliotecas compartilhadas que **ld.so** ou **ld-linux.so** irão procurar na máquina. Esse arquivo é sempre gerado e atualizado por **ldconfig** baseando-se no arquivo **/etc/ld.so.conf**.