Dr Linux

TELEVISION DOCTOR AND KERNEL PROBLEMS

MARIANNE WACHHOLZ

Dr. Linux

Complicated organisms, which is just what Linux systems are, have some little complaints all of their own. Dr. Linux observes the patients in Linux newsgroups, issues prescriptions here for the latest problems and proposes alternative healing methods.

Shared Libraries or jointly used *libraries* contain standard functions which are used by many programs, such as output to the screen output. They are loaded once, when a binary program needs them, and then they can also be used by other programs.

Whenever new Linux distributions are released, the question crops up repeatedly in the newsgroups: "What does char-major-[1...255] actually mean?" The answer to this and other questions can be found in the following article.

Which library?

I wanted to install a new Netscape on my system, but I keep getting error messages about missing **Shared Libraries**:

```
$ ./netscape-installer
error while loading shared libraries: cannot
open shared library: cannot load shared object
file: No such file or directory
```

How can I find out which libraries are to be used?
Dr. Linux: When users have problems with missing shared libraries, the following problem scenarios are possible:

- You are installing a program version which is so new that the required library version is not (yet) present in the system.
- On the other hand the program could be so outdated that you cannot find the associated library versions on the current system.
- It can also happen that the missing library is not in fact installed.

To find out which libraries are missing, use the command *ldd program*. When you do so, it is best to specify the program which will not start with path name.

If you address the *ldd* query to the file *./netscape-installer*, however, the result does not lead immediately to the goal:

```
not a dynamic executable
```

So let's take a more systematic approach. A look into the Installer directory gives the following information:

```
perlemaxi:~/netscape-installer > ls -l
total 172
-rw-r–r– 1 perle users 4725   Nov 22 2000
README
-rw-r–r– 1 perle users 7671   Dec  2 2000
config.ini
-rw-r–r– 1 perle users 2206   Dec  2 2000
installer.ini
-rwxr-xr-x 1 perle users 19308  Nov 22 2000
license.txt
-rwxr-xr-x 1 perle users 1521   Nov 22 2000
netscape-installer
-rwxr-xr-x 1 perle users 126032 Nov 22 2000
netscape-installer-bin
```

First, all we want to know is what kind of files are involved. For this we need the command *file*, which identifies file types and immediately outputs the result on the command line. As you can imagine,

the call is simple:

```
perlemaxi:~/netscape-installer > file↵
netscape-installer
netscape-installer: Bourne shell script text
```

So this file is not a binary program, but a shell script, which just starts the actual, compiled Netscape code. This is tucked away in *netscape-installer-bin*, which the command *file* confirms:

```
perlemaxi:~/netscape-installer > file↵
netscape-installer-bin
netscape-installer-bin: ELF 32-bit LSB
executable, Intel 80386, version 1,
dynamically linked (uses shared libs),
stripped
```

The fact that there is a script hiding behind executable files, and that the content of the script can be viewed with any ASCII editor, is a common occurrence. But in the listing of a directory it is not immediately completely clear whether a script or a binary program is involved, since file name extensions such as *.bat* for Linux scripts, are unusual. It is only sometimes that programs written in the script language Perl are marked with *.pl* and shell scripts with *.sh*. It now remains to demonstrate what the command *ldd* outputs when it is applied to a compiled program:

```
perlemaxi:~ > ldd ~/netscape-↵
installer/netscape-installer-bin
   libgtk-1.2.so.0 => /usr/lib/libgtk-1.2.so.0
(0x40025000)
   libgdk-1.2.so.0 => /usr/lib/libgdk-1.2.so.0
(0x40155000)
   libgmodule-1.2.so.0 => /usr/lib/libgmodule-
1.2.so.0 (0x4018c000)
   libglib-1.2.so.0 => /usr/lib/libglib-
1.2.so.0 (0x4018f000)
   libdl.so.2 => /lib/libdl.so.2 (0x401b4000)
   libXext.so.6 => /usr/X11R6/lib/libXext.so.6
(0x401b8000)
   libX11.so.6 => /usr/X11R6/lib/libX11.so.6
(0x401c6000)
   libnsl.so.1 => /lib/libnsl.so.1
(0x402a9000)
   libutil.so.1 => /lib/libutil.so.1
(0x402bf000)
   libresolv.so.2 => /lib/libresolv.so.2
(0x402c2000)
   libm.so.6 => /lib/libm.so.6 (0x402d5000)
   libstdc++-libc6.1-1.so.2 =>
/usr/lib/libstdc++-libc6.1-1.so.2
(0x402f4000)
   libpthread.so.0 => /lib/libpthread.so.0
(0x4033c000)
   libc.so.6 => /lib/libc.so.6 (0x40352000)
   libXi.so.6 => /usr/X11R6/lib/libXi.so.6
(0x4046f000)
   /lib/ld-linux.so.2 => /lib/ld-linux.so.2
(0x40000000)
```

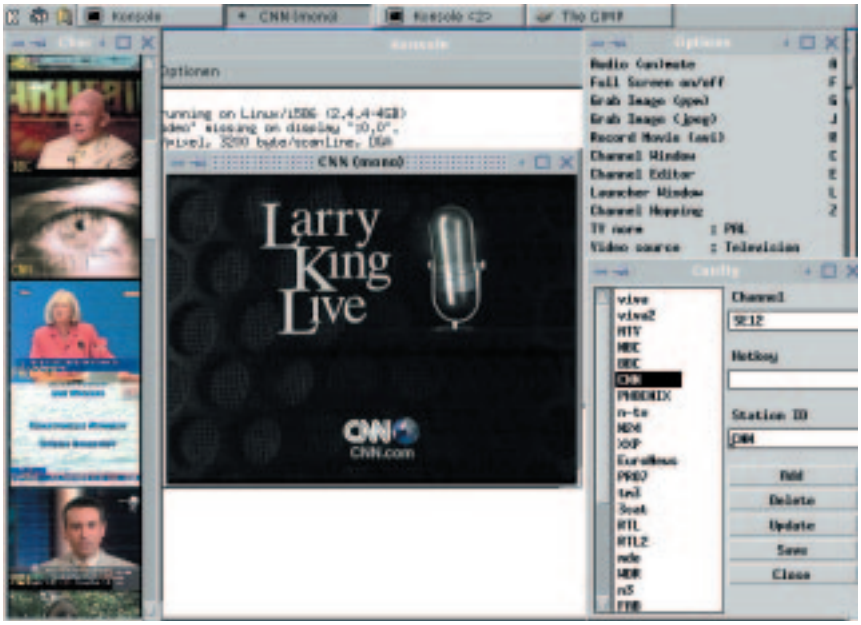

Figure 1: Xawtv in action

For example, if your system lacks *libm.so.6*, then *ldd* issues the message *=>not found* instead of the path to the library:

```
[...]
   libresolv.so.2 => /lib/libresolv.so.2
(0x402c2000)
   libm.so.6 => not found
   libstdc++-libc6.1-1.so.2 =>
/usr/lib/libstdc++-libc6.1-1.so.2
(0x402f4000)
[...]
```

I've got square eyes

I am using *Xawtv* to watch TV on my Linux system. Now I'd like to display the TV programmes in a specific sequence in the *channel window*. (This is the window which presents the available programmes for selection by mouse click.) Using the

Comment out: In scripts and many configuration files there is the option of having lines from the program being read in to be ignored by placing the # symbol at the start of a line. This means you can insert explanations into a file, without affecting its function. This additional information remains visible to humans, but when the file is evaluated by the computer it is ignored.

Listing 1: Extract from *~/.xawtv*

```
[...]
[defaults]
norm = PAL
input = Television
capture = over

[...]

[BBC]
channel = SE14

[CNN]
channel = SE12

# Comments preceded by a #
# are ignored.
#
#[DSF]
#channel = S23

[PHOENIX]
channel = SE19

[n-tv]
channel = SE5

[N24]
channel = S25
[...]
```

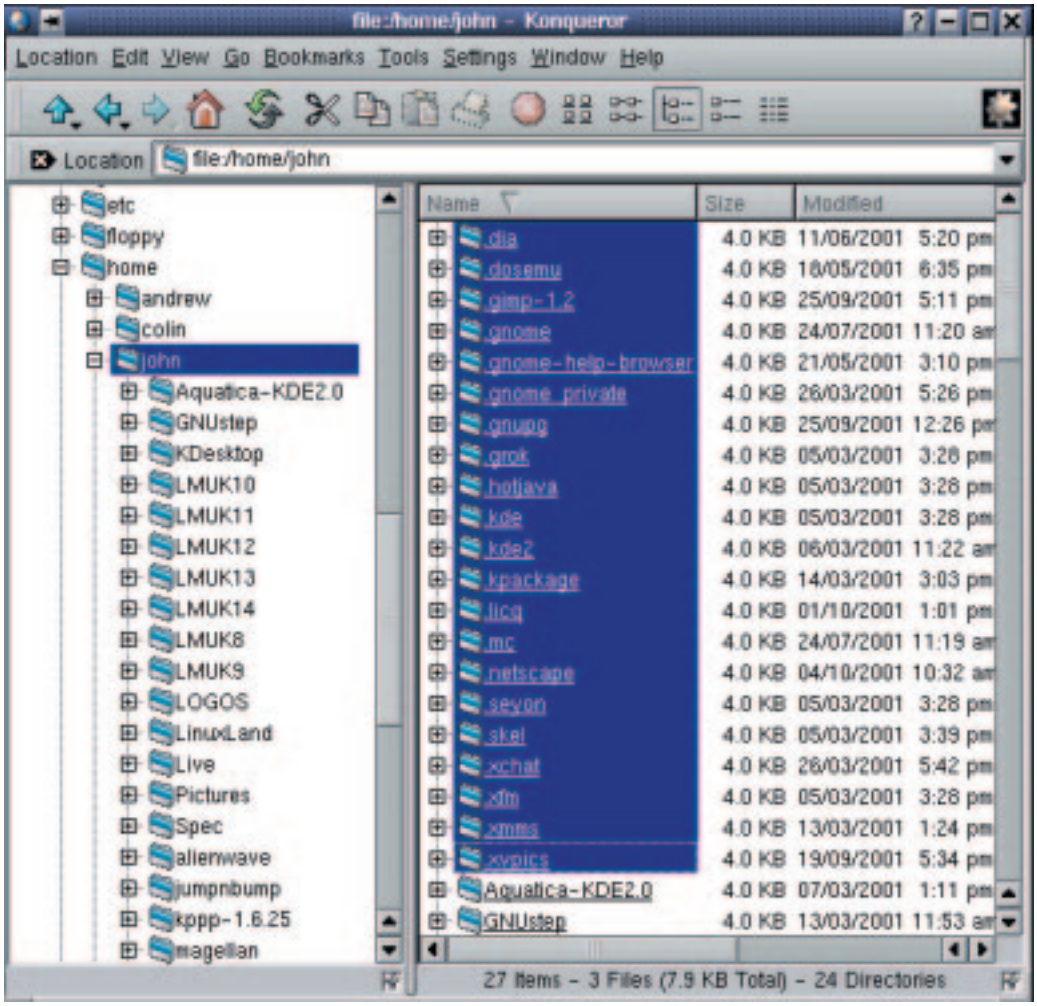

Figure 2: Konqueror displays hidden files on the right, but not on the left

channel editor this is a tedious task, as all details such a channels and broadcaster names have to be entered repeatedly. I can't find a configuration file named *Xawtv.config* or similar in the system.

Dr. Linux: When a Linux user has individually configured some piece of software, the associated configuration file is (almost) always to be found in his home directory. Normally such files and directories are made as *dot-files*. These are files/directories with a dot in front of their name (.). They are also known as *hidden files*, as in file managers, but also by *ls* they are not displayed without a bit more fuss. If you enter on the command line *ls* ("*l*ist") with the option *a* as in *a*ll, you will also be shown the dot files. The flag *l* also delivers a *l*ong output, providing additional information.

```
perlemaxi:~ > ls -al
total 1876
drwx—  — 44 perle users 4096 Jul  2 16:25 .
drwxr-xr-x  7 root  root  4096 Jun 26 14:11 ..
[...]
-rw-r—r—  1 perle users 1090 Jul  2 15:06
.xawtv
[...]
```

Obviously all file managers have the option for showing the hidden files. In the Konqueror of KDE 2.1.x for example this is done via *View / Display hidden files* (Figure 2).

The configuration file for *Xawtv* is simply called *.xawtv* and can be loaded into any editor. Its structure is simple and clear. Following a few configuration instructions are the broadcaster and place of transmission which you have already installed, one under the other. By using editor functions such as *Cut* and *Insert* you can very quickly change the programme sequence. And this also gives you the opportunity to **comment out** all "superfluous" channels (Listing 1 shows an example.)

If you want at some point to reactivate it, simply delete the comment symbol (#), and the channel is available, without you having to start a tedious sort process or having to configure Xawtv anew.

Modules: Drivers which are only loaded as necessary into a (modularised) kernel. The advantage of such a kernel compared with the *monolithic kernel*, which has all drivers embedded, is that only the drivers needed are loaded at run time and do not occupy any memory space when not in use.

Sources: Often called *source code*, *source text* or just *sources*. The text written by a programmer (in a programming language such as C++). This text, which is readable by humans, is only turned into a binary program or a file which is executable by computer by "translation programs" called *compilers*.

Character-Device: A character device, also known as a *character-oriented device* is read and written *sequentially* (in sequence), therefore the inputs and outputs are done **byte**wise. A serial interface is for example a character device.
Byte: Memory is divided into memory locations containing either the value 0 or 1. Such a location and/or the data stored in it is called a *bit*. Several bits can be combined into units such as a *byte*, *word* or *long word*: for example, one byte corresponds to eight bits.
Block-Device: On this kind of *block-oriented device* data is written block-wise and/or also read from it, with the device determining how big these blocks are. Hard disks or diskette drives are typical examples of block-oriented devices. You can access these devices by mounting them, with the *mount* command, in the file tree and later unmount them with *umount*.

Modular Problems

I keep getting the error message

```
modprobe: Couldn't find module char-major-108
```

Unfortunately I don't know how I can correct this error, as I don't even know what *char-major-number* means. Is there an explanation of this somewhere?
Dr. Linux: This kind of message tends to crop up following a system crash or after a faulty update when booting and in program calls. What it means is that the kernel **module** for *PPP* ("*P*oint to *P*oint *P*rotocol") was not found and therefore was not loaded.

The documentation on this can be found in the file */usr/src/linux/Documentation/devices.txt* and is part of the kernel documentation. This is only present on your system if you have installed the **sources** of the kernel on your system. Since these are extremely large, they are not usually automatically included in the system in a standard installation.

The kernel sources for your current system can be found, in common distributions, on the installation medium. From there, depending on the system they can be installed later with *YaST* (SuSE), *gnorpm*, *rpm* or *apt-get* (Debian). On the Internet you will find kernel archives packed as *tar.gz* or *tar.bz2* e.g. at *http://www.kernel.org/*. Anyone who does not have to worry about time spent online can also look up the latest version of the documentation on the website *http://www.kernel.org/pub/linux/docs/device-list/devices.txt*. They can also be downloaded separately by FTP (*ftp://ftp.kernel.org/pub/linux/docs/device-list/*). The file *devices.txt* lists which device is hidden behind a certain number. The document is in English, but very clear, because it is in the form of a list. The type is listed after the number in the first column: *char* stands for **Character-Devices** and *block* for the **Block-Devices**.

Patience is my middle name

It happens from time to time that I mistype my password. Then it takes far too long for a new login prompt to appear. Can this waiting period be cut?
Dr. Linux: With systems which want to offer a high level of security, the time-out to the next login prompt can certainly be very long. The time an evil-doer needs to try out various passwords thereby increases considerably. The Superuser is the one who defines how high the security requirement is for any specified computer. In the file the line

```
FAIL_DELAY number
```

stipulates how many seconds the pause between two attempted logins should last.

Listing 2: Extract from the file *devices.txt*

```
[...]
108 char   Device-independent PPP-interface
           0 = /dev/ppp         Device-independent PPP-interface

   block  Compaq Next Generation Drive Array, fifth controller
           0 = /dev/cciss/c4d0   first logic drive, whole diskette
          16 = /dev/cciss/c4d1   second logic drive, whole diskette
[...]
         240 =/dev/cciss/c4d15   16th logic drive, whole diskette

         The partitions are treated as Mylex DAC960
         (see major number 48), if one ignores the upper limit of 105
         possible partitions.
[...]
```