

Problems And Solutions In Linux

by Brian Long

Introduction

Here are a selection of the “problems” I’ve encountered whilst getting the hang of basic Linux usage along with the solutions I have found. These issues were encountered trying out a copy of RedHat Linux 6.0, although I’m hoping the commands referred to are generic. When I get my new machine I’ll be trying out Mandrake 7.1, SuSE 7.0 and Corel Linux 2nd Edition, so I guess I’ll know when that happens.

Bear in mind that I am still a complete newbie to Linux and haven’t got anywhere near trying to access networks, email, the Internet, a modem, printers or my sound card yet. These are all challenges I still have left for future.. er.. enjoyment. If I find anything else I feel might be useful, I’ll pass it along at a later date.

Linux Keystrokes

For those still coming to terms with Linux (among whom I include myself), here is a list of *possibly* useful keystrokes to help you along.

- Tab does auto-completion on file and path names.
- Ctrl+Scroll Lock shows state information for any running processes.
- Shift+Scroll Lock shows current memory information.
- Alt Gr+Scroll Lock shows current CPU register values.
- Ctrl+Z suspends a process so it can be sent to the background with the bg command (fg can be used to bring it to the foreground again) or killed by passing its process id (its PID) to the kill command.

You may or may not be aware that Linux allows you to start multiple consoles from the command prompt (you can login to each one and use it as a separate Linux session). Additionally, it is possible to start multiple sessions of X Windows, each of which uses a different virtual display, available through consoles 7 and higher. From the Linux shell prompt, you can use:

- Alt+F1 to Alt+F6 to switch between consoles 1 to 6.
- Alt+F7 to Alt+F12 switches between consoles 7 to 12, which are used for virtual displays (first display used goes on console 7, the second goes on console 8, etc. regardless of the actual virtual display number).
- Alt+↵ switches to the previous console.
- Alt+Ⓢ switches to the next console.
- Alt+Print Screen switches to the last console (or so I have read, but I cannot get it to work).

From inside X Windows, you have:

- Ctrl+Alt+Fn switches to console *n*. Once at a Linux shell, the shell keystrokes from above work. To get back to the first X session, switch to console 7.
- Ctrl+Alt+NumPlus switches to the next available configured screen resolution.
- Ctrl+Alt+NumMinus switches to the previous available configured screen resolution.
- Ctrl+Alt+BkSp kills the X server instantly and returns to the shell.
- In the Gnome desktop, Alt+Fn goes to virtual display *n*.
- In the KDE desktop, Ctrl+Fn goes to virtual display *n*.

Device names in Linux

Items of hardware, such as your mouse, monitor, hard drives, CD-ROM drives and so on are all accessed as devices in Linux. Devices are accessible through items in the /dev directory. You will inevitably need to refer to various devices in your time with Linux so here is a short summary of some that I have needed to refer to.

The mouse is normally accessed through /dev/mouse. The first floppy disk is accessed through /dev/fd0 (the second would be /dev/fd1).

A non-SCSI disk controller can serve several devices (I think up to four). The first one (often a hard drive) is referred to with /dev/hda, the second one (in my case a ZIP drive) with /dev/hdb, the third (a CD-ROM for me) with /dev/hdc and the fourth (a second CD-ROM) with /dev/hdd. SCSI drives have a prefix of sd instead of hd.

If one of these devices supports multiple partitions, these have their own device names. The first partition on a hard disk accessed through /dev/hda is called /dev/hda1, the second partition is /dev/hda2 and so on.

Accessing Windows drives from Linux

For newcomers like myself, even a task as simple as accessing the old drive C: can prove tricky until you know how. To access other partitions on your hard disks, you must *mount* them. This process can be automated through the linuxconf tool. In the Config tree, choose File systems, then Access local drive. From here you can mount drives by entering the device name for a partition, choosing the partition type and specifying the mount point (convention dictates that the mount point is placed in the /mnt directory).

For example, I access my C: drive by choosing the /dev/hda1 partition, indicating that it is a vfat drive and specifying a mount point of /mnt/drivec. After asking it to be mounted I can then list the files in my Windows directory with:

```
ls /mnt/drivec/windows
```

Multi-booting Linux and Windows from the NT boot loader

An email exchange between myself and Angus Jack regarding multi-booting Linux and Windows from the NT boot loader was printed in the last UK-BUG Magazine. I wish I'd known in advance as I have learned additional information on the subject since sending that mail to Angus.

To recap, the problem is how to get the Windows NT boot loader to recognise a bootable Linux partition and add it to its boot options menu (whose details are stored in BOOT.INI). The old message suggested using BootPart by G. Volland which can make a file containing the Linux partition boot sector and add a reference to it into BOOT.INI. The downside is that the boot sector contains an advert for BootPart which shows up when you choose the Linux partition (the ad is written out before the LILO prompt).

Not content with this situation I investigated further and learnt that the Linux dd command can also make a file containing the Linux boot sector, with no associated surreptitious ads. It relies on knowing as which device name the Linux boot partition is referred to by Linux (in my case it is /dev/hda8).

You tell dd to read one 512 byte block from the beginning of the boot partition (which will be the boot sector) and write it to a file name of your choice, perhaps in C:\ (which is mounted as /mnt/drivec in my case). So, for example:

```
dd if=/dev/hda8 of=/mnt/drivec/BootSect.Lnx
    bs=512 count=1
```

A reference to this boot sector file can then be added as another entry in BOOT.INI using any text editor, such as Notepad. For example:

```
[operating systems]
C:\BootSect.Win="Windows 98"
multi(0)disk(0)rdisk(0)partition(5)\WINNT="Windows
2000 Server" /fastdetect
C:\BootSect.Lnx="Linux"
```

Using your Windows TrueType fonts in X Windows

Your current Windows installation is replete with TrueType fonts, but a default X Windows installation doesn't have very many fonts available (at least that was what I found). However, you can give all your Windows TrueType fonts to X and it will use them, as long as you follow a number of steps.

Make sure the X font server (xfs) is running on your computer. You can ensure it is added to your runlevel with:

```
chkconfig --add xfs
```

Make a TrueType font directory:

```
mkdir /usr/X11R6/lib/X11/fonts/TrueType
```

Copy as many Windows TrueType fonts across as you like from your Windows font directory, using an appropriate file specification in the following file copy command, substituting the name you use to access your drive C: in place of /mnt/drivec:

```
cd /usr/X11R6/lib/X11/fonts/TrueType
cp /mnt/drivec/windows/fonts/*.ttf .
```

Generate a TrueType font directory file, which is used by the X server and the font server to find font files:

```
ttmkfdir > fonts.scale
```

Add the TrueType directory permanently to the font path with either:

```
chkfontpath --add /usr/X11R6/lib/X11/fonts/
                TrueType
```

or temporarily from within an X session with:

```
xset fp+ /usr/X11R6/lib/X11/fonts/TrueType
```

The permanent change adds a new comma-separated directory value into the catalogue entry in the font server configuration file /etc/X11/fs/config.

The X font server needs to be restarted with:

```
/etc/rc.d/init.d/xfs restart
```

When X Windows restarts, the TrueType fonts should be available, according to the information I have found. However I also found that this isn't necessarily the case unless you add another FontPath entry into the X configuration file (/usr/X11R6/lib/X11/XF86Config). To do this, load the file into a text editor, locate all the other FontPath entries and add one more to the list using the same layout, but specifying the directory created above.

An alternative (and perhaps preferable) option to copying the font files onto your Linux partition is to leave all the TrueType fonts where they are and make the font scale file and font directory file in the Windows Fonts directory itself.

Starting X Windows

When configuring your X server, you probably found that it claimed to support 16 bit per pixel (16 bpp) and maybe higher values. If you specified screen modes that included several colour depths (for example 8 bpp, 16 bpp and 24 bpp) you will find that X will default to 8 bpp (256 colours). To make X run at 16 bpp, use:

```
startx - -bpp 16
```

To start X windows without tying up one of the consoles you can start it as a background process. The graphical output will still appear in console 7 (use Alt+F7 to get there), but your prompt comes straight back. To run any process in the background, suffix the command line with an ampersand, for example:

```
startx - -bpp 16 &
```

Getting X Windows to display at the correct resolution

This one seems to pose a problem for many people, and certainly did for me before I took the bull by the horns. The specific problem I suffered from was that even though I had run the Xconfigurator program apparently successfully, X refused to display at my favourite resolution. It could happily handle 640x480 and 800x600 but absolutely refused to perform at 1024x768.

Since I had told Xconfigurator that I wanted to use all 3 resolutions, it defaulted to starting at 640x480, using a virtual 1024x768 desktop (really frustrating, but not the actual problem). I could use Ctrl+Alt+NumPlus to switch

to 800x600 successfully, but switching to 1024x768 caused the screen to go black and everything to die (Ctrl+Alt+BkSp failed to terminate X). Actually I suspect that Linux was fine and it was my monitor that has been sent into a coma.

I was tempted to try the xvidtune application, which came with large warnings about how it might throw my monitor off a tall roof, or damage it in other severe ways, but unfortunately it proves useless in this scenario. xvidtune seems only capable of tweaking a video mode that already works.

Then I found xf86config, which looked like it would sort the problem out. xf86config doesn't try probing for any information. It relies on you telling it everything it needs to know about the mouse, keyboard and screen. Aha!

Before running this application, you should make sure you know what type of connection is used by your mouse, the details of your video card (including how much RAM it has onboard) and lots of technical information about your monitor. I found the technical details about my monitor from a Web site that seems to specialise in storing this (otherwise useless, surely?) information, <http://members.nbci.com/linuxdaddies>.

By the time I had finished entering this information, I was confident of success, but it was misplaced confidence in this case. Still at least by specifying all the hardware details, I'm getting to the stage of being in control of some parts of the system.

In the end I had to edit the X configuration file (/use/X11R6/lib/X11/XF86Config) myself (after making a backup copy of course). I found the *Monitor* section that described my monitor (identified by the vendor name and model name I had previously entered) and noticed that for each resolution I wanted, there were several mode lines that each specified different settings. Additionally, there were many mode lines for resolutions that I had no interest in. These unnecessary lines were deleted immediately.

Next I commented out every 1024x768 modeline bar the first one, started X and tried the resolution to see if it would work. I realised that again, my monitor was possibly being put at risk, but let's face it, it already had been tested by the settings that Linux had assumed would work. The first few gave a similar dire failure, but eventually I found the line that worked. All the bogus ones were then deleted.

I also removed all the bogus versions of the 640x480 and 800x600 lines, but not by trial and error. This time, xvidtune did help. xvidtune displays the same values as listed in XF86config so all it took was a switch to each mode to work out which mode line was being used.

The final job was to make X start in 1024x768 rather than 640x480. This was done by changing the order of the values in the *Modes* entry in the *Screen* section that relates to my video card. The default resolution should be first in the list. You can also change the default colour depth (how many bits per pixel, or bpp, will be used in the screen mode) with the *Depth* entry in the *Display* subsection.

The configuration file supports several *Display* subsections, each one describing modes supported for a given colour depth. I find it best to keep just one of these sections, which ensures that X starts up with the colour depth specified in that section. This saves you having to use the extra command-line parameters described above.

Shell issues

I have only briefly touched the Bash shell so far, but was pleased to find that ~ is a shorthand way of referring to the logged in user's home directory, for example this command changes the current directory to the home directory:

```
cd ~
```

Links

Clearly, there is an ocean of information about Linux on the Web. Having done a bit of toe-dipping, I have bumped into a few useful sites/pages. You can find my favourite Linux links on my Web site (<http://www.blong.com>) on the *Links* page. They include a couple of online books as well as several that help you get the hang of the Bash shell and the vi and emacs editors.



Brian Long is a freelance trainer and problem-solver specialising in Delphi and C++Builder work. He is currently filling in spare time trying to get to grips with Linux. Visit his Web site at www.blong.com or email him on brian@blong.com.