What is Linux?

History

Linux is built around the Linux kernel originally released by Linus Torvalds in 1991. It was developed further by volunteers and started to be used with software from the GNU project, a suite of programs that form a complete operating system. Linux continues to be maintained by communities of volunteers. It is an open source project, meaning it's source code is publicly available and can be downloaded for free.

GNU/Linux software is modelled on the Unix operating system, developed in the early 1970s by Dennis Ritchie and Ken Thompson at Bell Labs.

Where is Linux used?

Linux was originally a free operating system for PCs, but it quickly became popular on web servers, and in the 1990s became one of the most popular operating systems used by web hosting companies. It is often used with the Apache web server, MySQL database software and the PHP programming language. This combination is known as a LAMP stack.

Many mainframe manufacturers have also adopted Linux as their operating system of choice.

As processing power got cheaper, the processors used in devices like set top boxes and mobile phones got more powerful. Eventually they became powerful enough to run a full featured operating system like Linux. Linux was widely adopted by many electronics manufacturers because it is free, and its source code is easily available. Linux now runs on everything from mainframes to smart watches.

Linux Distributions

There are many different versions, or distributions, of Linux. Some distributions are managed and maintained by communities of volunteers, and some distributions are maintained by commercial companies.

Linux is free, but there are still some restrictions on what can be done with it. Linux is released under the terms of the GNU General Public License (GPL), a long and complex document which essentially says you're not allowed to pass off GNU code

as your own work, and if you modify GNU code, you need to publish your modifications.

Not all software that runs on Linux is free. Many companies sell commercial software that runs on Linux, and this software is usually subject to much stricter license restrictions than the GPL. This software usually cannot be copied or freely distributed. This is equally true in embedded devices where electronics manufacturers have developed proprietary software that runs on Linux.

Raspbian

Debian was released in 1993. It's maintained by volunteers, and has a reputation for being very stable. The Raspberry Pi Foundation use Debian as a base for their Linux distribution, Raspbian.

Raspbian is built on an ARM port of Debian and includes the LXDE desktop, along with educational tools like Scratch and IDLE. The raspi-config program was added to make it easier to manage system settings. Raspbian includes libraries for controlling the Raspberry Pi's General Purpose Input/Output pins.

You can download Raspbian here: http://www.raspberrypi.org/downloads.

Comments

Files and directories in Linux

The root of the Linux file system is the '/' directory. All other drives, directories and files are contained in this directory. The Linux directory structure follows conventions laid down by the Unix directory structure, where different types of files put in specific directories:

- /bin - binaries/exectuables
- /boot - boot parameters and kernel
- /dev - devices
- /etc - configuration files
- /home - users' home directories
- /lib - libraries and system modules
- /media - removable drives are mounted here
- /mnt - permanently attached devices are mounted here
- /proc - virtual directory tree containing information about the operating system
- /root - root user's home directorydirectory

- /tmp - temporary files
- /usr - level programs, libraries
- /var - dynamic data like logs, web site content, print jobs

Every user has their own home directory. The default user on a Raspberry Pi is named pi, so the home directory for this user is /home/pi.

When packages are installed, their configuration files are place in /etc. If you have installed Apache and Samba, you will find their configuration files in /etc/apache2 and /etc/samba.

The /proc directory contains virtual files that represents operating system information in files. For example, this command will display information about the CPU:

```
$ cat /proc/cpuinfo
Processor       : ARMv6-compatible processor rev 7 (v6l)
BogoMIPS : 697.95
Features : swp half thumb fastmult vfp edsp java tls
CPU implementer  : 0x41
CPU architecture: 7
CPU variant      : 0x0
CPU part : 0xb76
CPU revision     : 7

Hardware : BCM2708
Revision : 000d
Serial           : 00000000da09f419
```

The cat command simply prints the contents of a file in the terminal. The file cpuinfo is Linux's representation of the CPU that it's running on.

Disk drives

Disk drives aren't represented in Linux in the same way as they are in Windows. In Linux, a computer's peripheral devices are generally represented as files in /dev. You can see your disk drives in the /dev directory by opening a terminal (double click on the LXTerminal icon on the desktop) and typing this command:

```
$ sudo fdisk -l
```

You should see output that looks like this:

```
Disk /dev/mmcblk0: 3965 MB, 3965190144 bytes
4 heads, 16 sectors/track, 121008 cylinders, total 7744512 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00014d34

        Device Boot      Start         End      Blocks   Id  System
/dev/mmcblk0p1            8192      122879       57344    c  W95 FAT32 (LBA)
/dev/mmcblk0p2          122880     7744511     3810816   83  Linux
```

This is information about the SD card that your Pi uses as a disk drive. You can see the Raspberry Pi's SD card listed as /dev/mmcblk0, and the two partitions on it are listed as /dev/mmcblk0p1 and /dev/mmcblk0p2.

Note that you can't access the files on your SD card via /dev/mmcblk0p1 or /dev/mmcblk0p2. These files are used by Linux device drivers to interact with the card.

Mounting drives

To access the files on the card, Linux must 'mount' it. This means Linux must open the disk and link the disk's directories with the rest of the file system. The directories are attached to an existing directory. For example, I have created a directory on my Pi called usbhdd in /media. When I connect a USB hard disk to my Pi and mount it, all the files and folders can be accessed in /media/usbhdd as if the disk was just a subdirectory in /media.

Mounting drives in this way makes it possible for Linux to have a single, continuous file system seamlessly spanning multiple drives.

File System Formats

Linux uses different disk drive formats for storing files than Windows. On Windows, disks and SD cards are formatted with the FAT32 file system or NTFS. On Linux, the

most common file system is ext4. Linux can read drives formatted with Windows file systems, but Windows can't access drives formatted with Linux file systems.

Absolute and relative paths

When you specify the location of a file, you can use its complete path. If I create a file in my home directory called myfile.txt, its complete path is /home/pi/myfile.txt. This is known as an absolute path.

It is also possible to use a relative path to access this file. When I log into Linux as pi, the default working directory is /home/pi. I can reference my file as ./myfile.txt. Unix systems use a single dot to refer to the current working directory, so a path starting with './' is relative to the current directory. Typing this command lists the contents of the current directory:

```
$ ls ./
```

Two dots are used to refer to the next directory up in the directory tree. If the current working directory is /home/pi, then typing this command lists the contents of /home:

```
$ ls ../
```

The '~' character refers to the current user's home directory. If I use the 'cd' command to change the current working directory to /etc (for example), this command will list the contents of my home directory:

```
$ ls ~
```

Hidden files and folders

File names and directory names that start with a dot are hidden. On a fresh install of Raspbian, if you type ls ~, there are only a couple of directories. If you type ls -a ~, the ls command will list everything in the home directory, including hidden files and folders.

See also: Connect your Raspberry Pi to a USB hard disk.

Comments

Users and Permissions

In order to use a Linux computer, the first thing you need to do is login. If you don't see a login screen when your Pi boots up, you've probably been logged in automatically. Each user has their own username and password. All the files owned by a user are marked with that user's name.

All files and directories have three sets of permissions:

- permissions for the owner of the file,
- permissions for users who are in the group that owns a file,
- permissions for all other users.

In each of these sets there are three basic permissions:

- read,
- write,
- execute.

You can view file permissions using the ls command with the '-l' option.

```
$ ls -l ./subfolder/
total 8
-rw-r-xr-x 1 pi pi 33 Nov  1 14:07 anexecutablefile.sh
drwxr-xr-x 2 pi pi 4096 Nov  1 14:13 tempfolder
-rw-r--r-- 1 pi pi 10 Nov  1 14:06 textfile.txt
```

The first character on each line is either a 'd' for a directory, or a '-' for files. The next three characters are the permissions for the owner of the file. The next three characters after the owner's permissions are the files group permissions, and the final three characters are show the permissions for other users.

These permissions are stored with each file and directory as metadata on the SD card in your Pi. The username of a file's owner is stored with its permissions.

The chmod command

The owner's permissions for anexecutablefile.sh are 'rw-' meaning user pi can read and write this file. This file is a script, so it needs to be given executable permissions:

```
$ chmod +x ./subfolder/anexecutablefile.sh
```

Now if I list the folder contents again, the output is slight different. This time there's an 'x' after the 'rw' characters on the line with anexecutablefile.sh:

```
$ ls -l ./subfolder/
total 8
-rwxr-xr-x 1 pi pi 33 Nov  1 14:07 anexecutablefile.sh
drwxr-xr-x 2 pi pi 4096 Nov  1 14:13 tempfolder
-rw-r--r-- 1 pi pi 10 Nov  1 14:06 textfile.txt
```

If necessary I can remove executable permission from this file using a '-' instead of a '+':

```
$ chmod -x ./subfolder/anexecutablefile.sh
```

The chmod can be used to set several permissions at once. This command makes sure that anexecutablefile.sh has read, write and execute permissions:

```
$ chmod +rwx ./subfolder/anexecutablefile.sh
```

The permissions on all the files and sub directories in ./subfolder can be set by using the -R option with chmod. If I wanted to remove write permissions for everything in subfolder, I could use this command:

```
$ chmod -w -R ./subfolder
```

Note, when the execute permission is applied to a directory, it just means that the users have the right to view the contents of the directory. It has no bearing on whether files in a directory can be executed. If I take away the executable permission

for subfolder and run the ls command again, this time the contents of subfolder won't be displayed.

Groups and other users

Groups of users can be created when several users need access to the same set of files. By default the user pi is part of a group called pi. I can grant privileges to files that allow other users who are in group pi to access them:

```
$ chmod g+rwx ./subfolder/anexecutablefile.sh
```

Now anyone in group pi can read, write or execute this file.

By default other users can read the files in subfolder, but can't write to them. To remove the read rights for other users, I can use this command:

```
$ chmod o-r ./subfolder/anexecutablefile.sh
```

Octal

Octal is a base-8 numerical system, where the highest digit in a number is 7. Octal numbers are often used to represent file permissions. An octal number can be expressed as a three bit binary number where the first bit represents the read permission, the second bit represents the write permission and the third bit represents the execute permission.

If a file has read and write permissions, then the first two bits would be true, and the third would be false, 110. If 110 is converted from binary to octal, its value is 6. If the executable permission were set, all three bits would be 1, so the octal value would be 7.

There are three sets of permissions (owner, group and other), so three octal digits can be used to represent all of a file's permissions. The first digit refers to the permissions for a file's owner, the second digit refers to the permissions for groups, and the third digit refers to permissions for others. The permission string -rw-r-xr-x could be represented in octal as 655.

If I wanted to use the octal notation to set executable permission of my script, I could use this command:

```
chmod 744 ./subfolder/anexecutablefile.sh
```

I can remove executable permission using this command:

```
chmod 644 ./subfolder/anexecutablefile.sh
```

I rarely use the octal notation as I prefer the symbolic notation described above. I find it easier to control specific permissions without changing other permissions unintentionally.

Root user

There is a user account called root. This user account has more privileges than other users, and is often used for system administration. The root user's home directory is /root.

Any user can assume root privileges with the sudo command. For example, if I login as pi and try to edit files in /etc, I won't be allowed to because user pi doesn't have the necessary permissions to edit files owned by root. This command will open a file in the nano text editor:

```
$ nano /etc/resolv.conf
```

The problem is that if I make changes, I won't be able to save them. User pi has enough permission to read this file, but not enough permission to write to it. If I use the sudo command to open a text editor, then I have root's permissions and I can save changes to the file:

```
$ sudo nano /etc/resolv.conf
```

It's important to be careful when using sudo as it's possible to accidentally lose important information or break configuration settings. It's best to back up configuration files before you edit them.