Debian System Admin (1)

The file system

Linux (and other POSIX-like operating systems) work with a single file hierarchy with a root folder `/`, although there may be different file systems mounted at different places under that. How files are organised in here are documented in the Filesystem Hierarchy Standard (FHS). Have a look with the command `ls /`:

`/bin` stands for binaries, that is programs that you can run. Have a look with `ls /bin`: there will be a lot of commands in here, including ls itself. Indeed you can find out where a program is with `which`, so `which ls` will show you `/usr/bin/ls` for example.

`/usr` is a historical accident and a bit of a mess. A short history is on this stackexchange question but essentially, in the earliest days,

- `/bin` was only for binaries needed to start the system - or at least the most important binaries that needed to live on the faster of several disk drives, like your shell.
- `/usr/bin` was where most binaries lived which were available globally, for example across all machines in an organisation.
- `/usr/local/bin` was for binaries installed by a local administrator, for example for a department within an organisation.

In any case, `/usr` and its subfolders are for normally read-only data, such as programs and configuration files but not temporary data or log files. It contains subfolders like `/usr/bin` or `/usr/lib` that duplicate folders in the root directory. Debian's way of cleaning this mess up is to make its `/bin` just a link to `/usr/bin` and putting everything in there, but in some distributions there are real differences between the folders.

If you have colours turned on (which is the default) you will see some files are green, but others are blue - this indicates the file type, green is an executable program, blue is a link to another file. Have a look with `ls -l /bin`: the very first character of each line indicates the file type, the main ones being `-` for normal file, `d` for directory and `l` for a so-called *soft link*. You can see where each link links to at the end of this listing. For example, `slogin` links to `ssh`. Other links point at files stored elsewhere in the filesystem -- you'll see a lot of references to `/etc/alternatives/`.

`/etc` stores system-wide configuration files and typically only root (the administrator account) can change things in here. For example, system-wide SSH configuration lives in `/etc/ssh`.

`/lib` contains dynamic libraries - windows calls these `.dll` files, POSIX uses `.so`. For example, `/lib/x86_64-linux-gnu/libc.so.6` is the C library, which allows C programs to use functions like `printf`.

`/home` is the folder containing users' home directories, for example the default user vagrant gets `/home/vagrant`. The exception is root, the administrator account, who gets `/root`.

`/sbin` (system binaries) is another collection of programs, typically ones that only system administrators will use. For example, `fdisk` creates or deletes partitions on a disk and lots of programs with `fs` in their name deal with managing file systems. `/sbin/halt`, run as root (or another user that you have allowed to do this), shuts down the system; there is also `/sbin/reboot`.

`/tmp` is a temporary filesystem that may be stored in RAM instead of on disk (but swapped out if necessary), and that does not have to survive rebooting the machine.

`/var` holds files that vary over time, such as logs or caches.

`/dev`, `/sys` and `/proc` are virtual file systems. One of the UNIX design principles is that almost every interaction with the operating system should look to a program like reading and writing a file, or in short *everything is a file*. For example, `/dev` offers an interface to devices such as hard disks (`/dev/sda` is the first SCSI disk in the system, and `/dev/sda1` the first partition on that), memory (`/dev/mem`), and a number of pseudoterminals or ttys that we will talk about later. `/proc` provides access to running processes; `/sys` provides access to system functions. For example, on some laptop systems, writing to `/sys/class/backlight/acpi_video0/brightness` changes the screen brightness.

The `/vagrant` folder is not part of the FHS, but is our convention for a shared folder with the host on Vagrant virtual machines.

Package managers

Linux has had package managers and repositories since the days when it was distributed on floppy disks. A repository is a collection of software that you can install, and can be hosted anywhere - floppy disk, CD-ROM, DVD or nowadays on the internet. A package manager is software that installs packages from a repository - so far, this sounds just like an *app store* but a package manager can do more. For one thing, you can ask to install different versions of a particular package if you need to. But the main point of a package manager is that packages can have dependencies on other packages, and when you install one then it installs the dependencies automatically.

To illustrate this, we're going to go on a tour of one kind of software with which you'll want to get very familiar: text editors that work in the console. Text editing is fundamental to a lot of system administration tasks as well as programming, and people often find that a familiar editor becomes a favourite tool.

Two console-based text editors are already installed in Debian: `nano` and `vim`.

`nano` is a basic text editor that works in the console, and is installed in most Linux distributions including the ones on seis and the lab machines, so you can use it to edit files remotely. You can type `nano FILENAME` to edit a file. The keyboard shortcuts are at the bottom of the screen, the main one you need is Control+X to exit (it will ask if you want to save, if you have unsaved changes). Nano is considered fairly friendly as console editors go.

`vim` is the 1991 improved version of the even older (1976) `vi` editor (which is also installed). It is a modal editor with a steep learning curve, but its extremely powerful editing grammar and widespread availability mean it regularly appears towards the top of lists of favourite text editors. If you want to get started with vim, I suggest you start by typing `vimtutor` at the commandline -- this opens vim with a file that guides you through basic vim usage.

Another console-based editor dating from the mid-70s is `emacs`, `vi`'s traditional Lisp-based rival. However, emacs is not installed by default: type `emacs` at the console and you will get `emacs: command not found`. You can install it with the command

```
sudo apt install emacs-nox
```

- `sudo` (superuser do) allows you to run a command as root, also known as the administrator or superuser. Depending on how your system is configured, this might be not allowed at all (you can't do it on the lab machines), or require a password, but on the Vagrant box you're using you are allowed to do this. It is good practice to use sudo for system adminstration instead of logging in as root directly, but if you ever really need a root shell then `sudo bash` gets you one - with `#` instead of `$` as prompt to warn you that you are working as root.
- `apt` is the Debian package manager.
- `install PACKAGE` adds a package, which means download and install it and all its dependencies (you'll see a list of these to confirm you want to install them -- you do).

We're installing `emacs-nox` because this is the version of emacs packaged for use from the console. If you tried to install just `emacs` then the package manager would identify that you need a graphical display manager to run emacs' GUI mode and install a lot more dependencies to enable that, which we don't need.

Now that emacs is installed, you can launch it with `emacs`, and from there you should see some introductory instructions, including how to access an emacs tutorial which will teach you how to use it. (If you want to exit emacs, Control-X followed by Control-C should do it).

Other popular editors include

- `mcedit`, a file editor which comes as part of the 'midnight commander' package, which you can install with `sudo apt install mc`. Launch the editor with `mcedit filename` and test it out (Alt-0 to exit).
- `tilde`, an editor with a GUI-esque menu system controlled through Alt-letter sequences. Install with `sudo apt install tilde`.
- `micro`, a simple editor somewhat like an advanced version of `nano`. Install with `sudo apt install micro`.

You can also find out information about packages with `apt info PACKAGE` -- try this for one of the above.

I suggest that you try out some of these editors, and figure out how you prefer to edit files. Some of these tools might require time investment to learn, but doing this early in your CS career could be a good decision.

Whichever editor you end up deciding to use, you probably won't need to keep all the alternatives installed. You can leave `nano` and `vim` installed, but for the other editors you've tried out above and decided you don't like, you can (and should) remove them from the system with `sudo apt remove PACKAGE`.

Update and upgrade

The repositories that you are using are recorded in `/etc/apt/sources.list`, have a look at this file with `cat` or `nano` to see where they are, then look up the sites in your browser. There are folders for different Debian versions and package index files for different architectures.

Two commands a system adminstrator should run regularly for security reasons:

- `sudo apt update` fetches the new package list from the repository. This way, apt can tell you if any packages have been updated to new versions since you last checked.
- `sudo apt upgrade` upgrades every package that you already have installed to the latest version in your local package list (downloaded when you do an `apt update`).

Lab machines

If you are running a virtual machine on the lab machines, then your virtual machine might not be around after the lab machine reboots or you log out and in again and end up on a different machine - as the notice when you log in tells you, the virtual machines are stored under `/tmp`.

It would be annoying to have to reinstall your favourite packages (like your chosen text editor) every time you log in to a different machine, so you should put them in your Vagrantfile and then `vagrant up` will do this for you automatically. The Vagrantfile already contains a line `echo`, you can put an `apt install PACKAGE` line below this, and can list as many packages as you like. There is no `sudo` here because when Vagrant is installing the system, it is running as root automatically.

- Unless it is `nano` or `vim`, add the package for your favourite text editor to this line so next time you rebuild the Vagrant machine, they are added automatically.
- Log out of your vagrant machine and do a `vagrant destroy` which removes the virtual machine. Then reload with `vagrant up` which will download and provision the box again.
- Log in with `vagrant ssh` and check that the editor is installed.