CLI Spells for the Raspberry Pi

From eLinux.org

This Page is a Quickstart for Command Line Interface on the Raspberry PI.--Brian 09:13, 20 April 2012 (UTC)
Back to Beginners Page (http://elinux.org/RPi_Beginners)

Contents

- 1 Why Do I Need the CLI?
- 2 Commands are just programs
- 3 General Syntax of Commands
- 4 Task $ cd. Navigating the file system using "cd"
- 5 Task $ ls. Listing the files and Folders in a particular place
- 6 Task $ chown Change Ownership of directories and files
- 7 Task $ chmod Change Access to directories and files
- 8 Task $ sudo. Invoking commands as an admin user
- 9 Task $ su - Becoming the admin user
- 10 Task $ touch. Create a new empty file
- 11 Task $ mkdir. Create a new empty directory
- 12 Task $ rm. Remove a file
- 13 Task $ rmdir. Remove a Directory
- 14 Task $ mount Connect to a device or Filing System
- 15 Some useful commands
- 16 References

Why Do I Need the CLI?

(CLI-Command Line Interface)

The Raspberry Pi operating systems that you can download for your SD card are all versions of Linux. A graphical user interface (GUI) for Linux is provided by the X Windowing System, together with a window manager. There are lots of different window managers, for example the one being used at time of writing with the Debian "squeeze" OS is LXDE. For some OS distributions, the windowing system is started up automatically when the machine starts up, but if not then you are in the Command Line Interface, which allows you to type commands and get information back. Most of us will immediately type "startx" to bring up the GUI, but even when you are using the GUI you can still get a CLI in the form of a terminal window. You can bring this up from the start menu; usually it is called something like X Terminal.

You need the CLI because there are many things that either can't be be done using GUIs, or are much easier using the CLI. In addition there are situations when you would rather not run the X Windowing system, because it takes up a lot of processor power and memory. Also a Raspberry Pi doing some task or other may well have no screen at all, and you may manage it remotely using a remote log in to a terminal.

Windows and Mac users are generally guided away from the CLI hence this Quick start Guide. The commands here have mostly been described and tested using debian linux which at the time of writing (April 2012) was the version of linux that was most active in the RPi world. It is not meant to advocate this distribution in particular. The author (Brian 09:27, 30 May 2012 (UTC)) is actually a Fedora user most of the time.

The quick answer to "why do I need the CLI?" then, is that the CLI is going to be a very useful tool for Raspberry Pi users.

Commands are just programs

When you type into the terminal you are running programs. Most of the commands listed here run programs that give you the ability to command the system to do something. When you add programs to your pi, you will be able to run more commands. This means that if a command you run and expect to work returns the message $ bash: foo: command not found...

It just means that you have not installed "foo" yet. Most of the commands described in this page are the kind of house keeping commands that come with bash. Each program has an original author who is acknowledged at the bottom of the man page.

To use the CLI you need to know commands and the default command set is contained in the "shell" you are using. There are Lots of shells out there and the one on this Raspberry Pi is Bash Wikipedia Bash Entry (http://en.wikipedia.org/wiki/Bash).

To find the syntax for a command do an Internet search for the command or find a Bash Quick Reference Card. In this wiki entry, each command has a link to the Wikipedia entry for that command.

If you want to know how to use commands without using the Internet then use man (short for manual).

```
$ man <command>
```

or

```
$ info <command>
```

from the CLI to be sent to a page of guidance.

The guidance given by "man" is sometimes a bit formal to the point that it could be said, "if you think you know a command then go to the command's man page in order to find out that you don't really".

General Syntax of Commands

Commands take the form

```
<Command> | -<Switches> | <Parameters> | <Target>
```

<> are often used in Syntax guides to indicate that the <> surround a place for a command rather than an actual command. | is used to denote OR or Optional so in the line above we read that a Command can be used on its own but may be followed by Switches (single letters, preceded by a hyphen that adjust what the command does), Parameters (Things that the command needs to know in order to work) and a Target (The thing that the command will be applied to).

e.g. "`ls -l /home/brian`" means List in Long Format (the `-l` switch) the directory (target) `/home/brian`

```
mount -F smbfs //workgroup;fred:foo@192.168.1.99/homes /mnt/net
```

Means use the username `fred` and password `foo` (parameters) to make the shared drive called `homes` on the Windows server at `192.168.1.99` (parameter) appear in the directory tree at the point `/mnt/net` (target) using the Server Message Block Filing System (the `-F` switch). (BTW, this assumes that you have the samba client software installed on your pi)

Task $ cd (http://en.wikipedia.org/wiki/Cd_%28command%29). Navigating the file system using "cd"

The file system in Linux is Hierarchical with nested directories. (Often called Folders) in a "Tree". The top of the directory structure is "`/`" and Directories underneath `/` are referred to using "Paths" just like urls in a web browser. To go to a particular place in the directory structure you use *`cd`*. *e.g.*

```
cd /
```

will take you to the top of the Directory Tree

To go to a particular place use cd followed by the location in the tree.

```
cd /home/brian/Documents
```

will take you to "`brian`'s" home directory `Documents` Folder. **File & Directory Names are Case Sensitive in Linux**

To go one Directory "Up" the tree then do

```
cd ..
```

Task $ ls (http://en.wikipedia.org/wiki/Ls). Listing the files and Folders in a particular place

```
$ ls
```

list all the file in the current directory.

```
$ ls -l
```

list all the file in the current directory and display the long version of the information about each file or directory. The output may be colour coded depending on the terminal preferences that you have set. `ls -l` tells you (amongst other stuff) the size, owner and security setting on each file. Other useful invocations are

```
ls -R
```

Recursive. i.e. list contents of sub-directories as well as this directory.

```
ls -A
```

Show "Hidden" Files. ie. Hidden files are file names that start with a dot which in Unix GUI file navigators are (by default) hidden.

Examples.

```
$ ls
An_Gott_und_meine_Mutter.mid   Domestic        Programming          Test.mid
An_Gott_und_meine_Mutter.mscz  Engineering     Quantum Physics      Tutoring
appliances                     FluidR3_GM.ins  School               Windaz
...
$ ls -l
total 336
-rw-rw-r--.  1 brian brian   2429 Apr  2 20:27 An_Gott_und_meine_Mutter.mid
-rw-rw-r--.  1 brian brian   4085 Apr  2 19:52 An_Gott_und_meine_Mutter.mscz
drwxrwxr-x.  4 brian brian   4096 Apr  2 20:38 appliances
-rw-rw-r--.  1 brian brian  10919 Apr  2 19:52 brotplot.odt
.....
```

Task $ chown (http://en.wikipedia.org/wiki/Chown) Change Ownership of directories and files

In the Long List above, the files are owned by Brian and are also in the Brian Group.

```
$ chown fred brotplot.odt
```

after executing `ls -l` will result in

```
-rw-rw-r--.  1 fred brian  10919 Apr  2 19:52 brotplot.odt
```

The file is still in group `brian` but is owned by `fred`

```
$chown foo:foo brotplot.odt
```

will result in

```
-rw-rw-r--.  1 foo foo  10919 Apr  2 19:52 brotplot.odt
```

The group has changed also. See Also `chgrp (http://en.wikipedia.org/wiki/Chgrp)`

Task $ chmod (http://en.wikipedia.org/wiki/Chmod) Change Access to directories and files

When long list files, the left hand column takes the form `xuuugggwww`. `x` is a letter describing what kind of thing the list entry is and can be various things but most commonly is `d`, `l` or `-`. `d` stands for directory, `l` for link - for a file.

The `uuugggwww` describe access rights for the owner (user), group and the world. (i.e. everyone not in the set owner or group). In each triplet, the order is `rwx`, `r` for ability to read, `w` for ability to write and `x` for ability to execute. If the `x`flag is not set then the file will not run as a program.

This line from above tells us that fred and the group called students both have read and write access where everyone else has read access only.

```
-rw-rw-r--.  1 fred students  10919 Apr  2 19:52 brotplot.odt
```

There are various ways of `chmod` use but probably the easiest is to consider the groups `u` for user-owner, `g` for group, `o` for other(everyone else) and `a` for all three.

Use chmod as follows

```
chmod o+w brotplot.odt
```

means add write access for the other group to brotplot.odt

```
chmod a-r brotplot.odt
```

means remove read access for the all three (user, group and other) from brotplot.odt

```
chmod u+x brotplot.odt
```

means add execute access for the user/owner to brotplot.odt

Task $ sudo (http://en.wikipedia.org/wiki/Sudo). Invoking commands as an admin user

This command introduces the SuperUser or root user. Many commands are only permissible to the root user and by invoking sudo before a command, you are saying "do the following command as the root user". When you do this, you will be asked for your password and if you have system permissions (commonly called being in the admin group) to be a root user then the command will be run. If you do not then a message will tell you so.

`root` is the name for the main administrator of a Unix like system. The default Debian distribution has no `root` passwd set and you do privileged commands by prepending the command with `sudo`.

Example

```
adduser brian
adduser: Only root may add a user or group to the system.

sudo adduser brian
# You now enter a sript that sets up a new user called brian
```

Anytime you get an "`Only root can ....`" message, try the `sudo` command.

There is another way of doing things as `root` user and that is by using

Task $ su - (http://en.wikipedia.org/wiki/Su_%28Unix%29) Becoming the admin user

which stands for "substitute user" (or "switch user") and as invoked here means "become the `root` user". From this point on, all commands run as the super user. Invoking this command will cause you to be asked for the `root` password. Some systems do not allow this to happen in a default install in which case you have to enable the root user. To enable the root user issue

```
passwd root
```

(You will have to enter a new passwd). You will be asked twice for the password and assuming you can manage to enter the same password twice, you now have a root user and you can issue

```
su -
```

When you enter passwords in Unix like systems, you will not see asterisks or anything at the prompt but the password is being entered nevertheless. If you habitually do everything as the `root` user, you shouldn't. Eventually you will do something both educational and disastrous. For example, you could invoke

```
rm -r -f /*
```

and delete all the files in the whole system.

```
$ rm -r -f ./*
```

would just delete the files in the current directory. You do not usually get warnings as `root` user. (Other than this one).

Task $ touch (http://en.wikipedia.org/wiki/Touch_%28Unix%29). Create a new empty file

Go to where you want the file to be and invoke

```
$ touch <filename>
```

Task $ mkdir (http://en.wikipedia.org/wiki/Mkdir). Create a new empty directory

Go to where you want the directory to be and invoke

```
$ mkdir <Directory Name>
```

Task $ rm (http://en.wikipedia.org/wiki/Rm_%28Unix%29). Remove a file

Go to where you want the directory to be and invoke

```
$ rm filename
```

Task $ rmdir (http://en.wikipedia.org/wiki/Rm_%28Unix%29). Remove a Directory

Go to where you want the directory to be removed. The directory will already need to be empty of all contents.

```
rmdir
```

If the directory is not yet empty

```
$ rm -r -f <Directory Name>
```

This will remove the directory, all its contents and also any sub directories and their contents.

Task $ mount (http://en.wikipedia.org/wiki/Mount_%28Unix%29) Connect to a device or Filing System

`mount` is the way in which you connect a Unix system to external devices. There is no "C" drive as in Windows. What happens in Linux (and FreeBSD/OS X) is that a device is mounted in the filing system somewhere and when you navigate to that place. The items offered by the device will appear at that point.

This is a complex command. The switches, parameters and target of the Mount command will vary according to the protocol of the system being mounted. Some things will automount. This is why, when you plug a SDcard into a modern Linux, the filing system will automatically pick it up.

Manual Mounting requires a "Mount Point". That means a directory which will be filled with the mounted device when it is mounted. Often, this is in the directory `/mnt/` somewhere.

Generally, first ensure there is a mount point, if not then create a directory at the point needed. e.g.

```
$ mkdir /mnt/netfolder
```

Then issue `mount` with the necessary switches, parameters and directories.

e.g. From above.

```
mount -F smbfs //workgroup;fred:foo@192.168.1.99/homes /mnt/net
```

Some useful commands

```
startx
```

If you are a bit nervous about the command line, this might be your favourite. It gets you escape to a nice graphical interface.

```
free
```

Show how much memory is available.

```
df /
```

Show how much disk space is used.

```
hostname -I
```

Show your IP address. Try this command with your network cable connected and disconnected to see the difference.

```
lsusb
```

Show what is plugged into the USB port. Try this command with your mouse connected and disconnected to see the difference.

References

Raspberry Pi

- V
- T
- E (http://elinux.org/index.php?title=Template:Raspberry_Pi&action=edit)

Startup	Buying Guide - SD Card Setup - Basic Setup - Advanced Setup - Beginners Guide - Troubleshooting
Hardware	Hardware - Hardware History - Low-level peripherals - Expansion Boards
Peripherals	Screens - Cases - Other Peripherals (Keyboard, mouse, hub, wifi...)
Software	Software - Distributions - Kernel - Performance - Programming - VideoCore APIs - Utilities
Projects	Tutorials - Guides - Projects - Tasks - DataSheets - Education - Communities

Retrieved from "http://elinux.org/index.php?title=CLI_Spells_for_the_Raspberry_Pi&oldid=163622"

Category: RaspberryPi
