# Navigating Through the File System

The cd command is generally used with a specific directory location or pathname, like this:

**jsoap@ubuntu:~$ cd /etc/apt/**

To quickly move up to the parent directory, the one above the one you are currently in, use the cd command like this:

**jsoap@ubuntu:~$ cd ..**

To return to your home directory from anywhere in the Linux file system, use the cd command like this:

**jsoap@ubuntu:~$ cd**

Use pwd to remind you where you are within the file system.

The ls command lists the contents of the current directory. It’s commonly used by itself, but a number of options (also known as switches) are available for ls and give you more information. For instance, the following command returns a listing of all the files and directories within the current directory, including any hidden files (denoted by a . prefix) as well as a full listing, so it will include details such as the permissions, owner and group, size, and last modified time and date:

**jsoap@ubuntu:~$ ls -la**

## Basic Linux Directories

The following table shows some of the top-level directories that are part of a standard Linux distro.

* / The root directory
* /bin Essential commands
* /boot Boot loader files, Linux kernel
* /dev Device files
* /etc System configuration files
* /home User home directories
* /lib Shared libraries, kernel modules
* /lost+found Directory for recovered files (found after a file check)
* /media Mount point for removable media, such as DVDs
* /mnt Usual mount point for local, remote file systems
* /opt Add-on software packages
* /proc Kernel information, process control
* /root Super-user (root) home
* /sbin System commands (mostly root only)
* /srv Information relating to services that run on your system
* /sys Real-time information on devices used by the kernel
* /tmp Temporary files
* /usr Software not essential for system, such as applications
* /var Variable data (such as logs); spooled files

## GLOB patterns (wildcard pathname matching)

## A shell glob is a special character that you can think of like a wildcard. For instance, you can use the ? symbol in place of any single character. Let’s say that I have a directory with five files, a.txt, b.txt, c.txt, d.doc, and e.tar:

**$ ls a.txt b.txt c.txt d.doc e.tar**

## If I wanted to see only the .txt files, I could type all of filenames by hand:

**$ ls a.txt b.txt c.txt a.txt b.txt c.txt**

## or I could use ? as a wildcard:

**$ ls ?.txt**

**a.txt b.txt c.txt**

## You can use multiple shell globs as well, so if I also wanted to match e.tar, I could type:

**$ ls ?.t??**

**a.txt b.txt c.txt e.tar**

## Probably one of the most useful shell globs is the \* symbol. This wildcard matches any number of characters, including zero. In this example, let’s say I had the following files in my directory:

**$ ls**

**aa.txt ac.txt a.txt b.txt c.txt d.doc e.tar**

## If I wanted to list only the files that begin with a, I could type

**$ ls a\***

**aa.txt ac.txt a.txt**

## Note that the \* symbol acts as a wildcard for a number of different characters in the filename. If I wanted to move these files to the /tmp directory, for instance, I could type

**$ mv a\* /tmp/**

## If, instead, I wanted to move all of the .txt files to the /tmp directory, I would type

**$ mv \*.txt /tmp/**

## Unlike with the ? wildcard, when I use \* I also match aa.txt and ac.txt.

## Grep, Pipes & Redirection

Grep is a command line text search utility.

**#grep apple fruitlist.txt**

Grep prints all lines containing apple from the file fruitlist.txt. Grep can also be used with pipes.

**# ls**

**anotherfile.txt somedir somefile1.txt somefile2.txt**

**# ls | grep some**

**somedir**

**somefile1.txt**

**somefile2.txt**

In the second command you can see that I piped the output of the ls command to grep, and then told grep to filter only text that contained the keyword some. If I had wanted to see only text that had the word file in it, I could type

**# ls | grep file**

**anotherfile.txt**

**somefile1.txt**

**somefile2.txt**

You can also chain multiple grep commands together using pipes.

## Network Configuration Files

The primary network configuration file is /etc/network/interfaces and contains the configuration for all networking devices on the system. The loopback interface always has the address 127.0.0.1. Generally, you don’t have to worry about configuring this interface because Ubuntu should do it for you, but if you did, here is what it would look like:

**auto lo iface lo inet loopback**

The auto line specifies that the lo interface should automatically be brought up at boot. The iface line defines the lo interface and creates its settings (in this case defining lo as a loopback device). If your server gets its network settings for eth0 via DHCP, then the following would be a valid configuration:

**auto eth0**

**iface eth0 inet dhcp**

If, on the other hand, your network settings are static, here is a sample configuration:

**iface eth0 inet static**

**address 192.168.1.10**

**netmask 255.255.255.0**

**gateway 192.168.1.1**

The fields are pretty self-explanatory. The first defines eth0 as a static interface instead of one that uses DHCP. Then the address of the interface is set (192.168.1.10), then the netmask (255.255.255.0), and finally the gate- way (192.168.1.1) address—the IP for the router the machine will use to access other networks.

In addition to /etc/network/interfaces, there are two other core files for network settings. The first is /etc/resolv.conf and the second is /etc/hosts. If your server gets its settings via DHCP, this file will automatically be set. If you have a static address and set name servers for this machine during the initial install, they will appear here. If you need to change the settings, the syntax is pretty simple, as this example shows:

**search mysite.com**

**nameserver 192.168.1.12**

**nameserver 192.168.1.13**

The first line is optional but configures the DNS search path to use. This way, if I wanted to access web.mysite.com, I could just reference web and the system would first try to get an address for web.mysite.com. The next lines define the IP addresses of name servers to use.

When a system first boots, the /etc/init.d/networking script will read /etc/network/interfaces and automatically bring up any interfaces configured to load at startup, but sometimes you might want to take down or bring up an interface manually. The ifup and ifdown scripts will respectively bring up and take down the interface you pass as an argument, so if I wanted to take down eth0, I would type ifdown eth0. Other important networking commands are the ifconfig, route and nslookup commands.

Type nslookup followed by a name or IP address to resolve, and nslookup will return the results of the DNS query:

**nslookup ubuntu.com**

**Server: 192.168.248.2**

**Address: 192.168.248.2#53**

**Non-authoritative answer:**

**Name: ubuntu.com**

**Address: 91.189.94.156**

## Commands you can use to search the file system

whereis command — Returns the location of the command

**whereis ls**

**/bin/ls**

whatis command — Returns a one-line synopsis from the command’s man page.

**whatis clear**

**clear - clear the terminal screen**

locate file — Returns locations of all matching file(s)

**locate index.html**

**/var/www/html**

Automatically built once a day by system so you may need to run the **updatedb** command with super-user privileges to manually start the building of the database if you make changes.

## Managing Files with the Shell

Here are some commands operations:

* **cat filename** — Outputs contents of filename to display
* **less filename** — Allows scrolling while reading contents of filename
* **mv file1 file2** — Renames file1 to file2 and may be used with full pathnames to move the file to a new directory at the same time
* **mv file dir** — Moves file to specified directory
* **cp file1 file2** — Copies file1 and creates file2
* **rm file** — Deletes file
* **rmdir dir** — Deletes directory (if empty)

Note that each of these commands can be used with pattern-matching strings known as wildcards or expressions. For example, to delete all files in the current directory beginning with the letters abc, you can use an expression beginning with the first three letters of the desired filenames. An asterisk (\*) is then appended to match all these files. Use a command line with the rm command like this:

**jsoap@ubuntu:~$ rm abc\***

## Package Management in Ubuntu

The administrator of every Ubuntu installation—servers and desktops— must learn the basic mechanics of package management. Each Ubuntu system stores a list of package repositories in /etc/apt/ sources.list. To update the system’s list of packages, you can run apt-get update. This command downloads the latest updated package lists for all repositories listed in your /etc/apt/sources.list file.

To install a package:

**#apt-get install most**

To remove a package:

**#apt-get remove most**

The most command bills itself as a replacement for less. Like its predecessors less and more, most is a file pager program.

Installing any new version of packages is as simple as running apt-get upgrade command, this command simply tries to upgrade all installed packages to their most recent versions.

**#apt-get update**

**#apt-get upgrade**

## Working with Compressed Files

Another file management operation is compression and decompression of files.

* **tar** — Creates, expands, or lists the contents of compressed or uncompressed file or directory archives known as tape archives or tarballs

Most of these commands are easy to use. However, the tar command, which is the most commonly used of the bunch, has a somewhat complex set of command-line options and syntax. This flexibility and power are part of its popularity; you can quickly learn to use tar by remembering a few of the simple command line options. For example, to create a compressed archive of a directory, use tar’s czf options like this:

**jsoap@ubuntu:~$ tar czf dirname.tgz dirname**

The result is a compressed archive (a file ending in .tgz) of the specified directory (and all files and directories under it). Add the letter v to the preceding options to view the list of files added during compression and archiving while the archive is being created.

To list the contents of the compressed archive, substitute the c option with the letter t, like this:

**jsoap@ubuntu:~$ tar tzf archive**

However, if many files are in the archive, a better command is

**jsoap@ubuntu:~$ tar tzf archive | less**

To expand the contents of a compressed archive, use tar’s zxf options, like so:

**jsoap@ubuntu:~$ tar zxf archive**

## Working with Permissions

Under Linux (and UNIX), everything in the file system, including directories and devices, is a file. And every file on your system has an accompanying set of permissions based on ownership. These permissions provide data security by giving specific permission settings to every single item denoting who may read, write, and/or execute the file. These permissions are set individually for the file’s owner, for members of the group the file belongs to, and for all others on the system. You can examine the default permissions for a file you create by using the touch command and then the ls command’s long-format listing like this:

**jsoap@ubuntu:~$ touch file**

**jsoap@ubuntu:~$ ls -l file**

**-rw-r--r-- 1 jsoap jsoap 0 2010-06-30 13:06 file**

In this example, the touch command is used to quickly create a file. The ls command then reports on the file, displaying the following (from left to right):

* **The type of file created** — Common indicators of the type of file are in the leading letter in the output. A blank (which is represented by a dash, as in the preceding example) designates a plain file, d designates a directory.
* **Permissions** — Read, write, and execute permissions for the owner, group, and all others on the system.
* **Number of links to the file** — The number one (1) designates that there is only one file, whereas any other number indicates that there might be one or more hard-linked files. Links are created with the ln command. A hard-linked file is an exact copy of the file, but it might be located elsewhere on the system. Symbolic links of directories can also be created, but only the root operator can create a hard link of a directory.
* **The owner** — The account that owns the file; this is originally the file creator, but you can change this designation using the **chown** command.
* **The group** — The group of users allowed to access the file; this is originally the file creator’s main group, but you can change this designation using the **chgrp** command.
* **File size and creation/modification date** — The last two elements indicate the size of the file in bytes and the date the file was created or last modified.

**The chmod, chgrp, chown commands**

* **chmod** command is used to change permissions on a file.
* **chgrp** command is used to change the group of a file or directory.
* **chown** command to change ownership of a file or directory to one or more users.

**chmod**

The chmod command allows you to change permissions on a file. There are a number of different ways to describe permissions for a file (type man chmod to see a full list), but one common way is to list **u**, **g**, or **o** for **user**, **group**, or **other** categories followed by a **+** or **-** sign, and then the permission to add or remove.

So if I wanted to remove write access for the user who owns a file, I would type **chmod u-w filename.** If I wanted to add read permissions for a file’s group I would type **chmod g+r filename**. To add read permissions on a file for other users on the system, I would type **chmod o+w filename**. Let’s take our **script.sh** file as an example:

**$ ls -l script.sh**

**-rwxr-xr-- 1 jsoap admin 2011-03-04 15:13 script.sh**

If I wanted to allow other users on the system to execute this file, I would type

**$ chmod o+x script.sh**

**$ ls -l script.sh**

**-rwxr-xr-x 1 jsoap admin 2011-03-04 15:13 script.sh**

In addition to u (user), g (group), and o (other), the chmod command also accepts **a** for all categories. If, for instance, you wanted to remove execute permissions for user, group, and other, you could type

**$ chmod a-x script.sh**

**$ ls -l script.sh**

**-rw-r--r-- 1 jsoap admin 2011-03-04 15:13 script.sh**

To undo what I just did, I would just change it to a +x:

**$ chmod a+x script.sh**

**$ ls -l script.sh**

**-rwxr-xr-x 1 jsoap admin 2011-03-04 15:13 script.sh**