File and Directory Commands

cat >*file* : put whatever is typed on keyboard into *file*, until CTRL-D is entered.
cat >>*file* : append what is typed on keyboard to *file*, until CTRL-D
cat *file* : display *file* on screen, all at once
cat *file1 file2* >*file3* : join *file1* to *file2*, put in *file3*
cd *x* : change to directory *x*
../ means up one level
~/ means home directory
cp *file1 file2* : copy *file1* to *file2*
cp -r *dir1 dir2* : copy *dir1* contents to *dir2*
grep *pattern file* : search for *pattern* in *file*, print those lines to screen
head -n *file* : show the first *n* lines of *file*
ln -s *file link* : create a link *link* to *file*
ls -la : list all files in long format, with permissions
mkdir *dir1* : create directory *dir1*
more *file1* : display *file1*, one screen at a time; type q to stop the display or space to advance
mv *file1 file2* : move *file1* to *file2*; used to rename
pwd : print working directory (current location)
rmdir *dir1* : remove directory *dir1* (when empty)
rm -r *dir1* : remove all files and subdirectories in *dir1* and delete directory [use with caution!]
tail -n *file* : display the last *n* lines of *file*
wc *file* : display the number of lines, words, and characters in *file*. Use options **-l**, **-w**, or **-c** to see only lines, words, or characters, respectively.

File Permissions and Ownership

chmod ***octal*** *file* : change the permissions on *file* to those specified by *octal*, where user, group, and world permissions are specified by the sum of:
4 = read (r), 2 = write (w), and 1 = execute (x). For example, the octal 754 specifies rwx permissions for user, r-x permissions for group, and r-- for world, shown as -rwxr-xr-- in ouput from a **ls -l** command.
chown *user group file* : change the owner and group membership of a file – allowed only for the owner of the file and root.

Network Commands

ping *host* : send a signal to remote computer *host* to see if it is connected to the network
wget *URL/file* : download *file* from remote computer host via *URL*, which can be either a FTP, HTTP or HTTPS connection.

Remote Computer Connections

scp *file user@host.address :dir* **:** securely copy *file* from the local computer to the remote computer *host*, with encryption of the file and password during transmission.
ssh *user@host.address* : securely connect to the remote computer *host* as *user*
ssh -p *x user@host.address* : securely connect to the remote computer *host* on port *x* as *user*

System Information

cal : show the current month's calendar
cat /proc/cpuinfo : display information about processors
cat /proc/meminfo : display information about installed memory resources and utilization
date : show the current date and time
df : display occupied and free space on local disks
free : display memory and swap usage
man *command* : display information for *command*, type q to exit the man page.
uname -a : display operating system information
whereis *app* : show location of *app*, if installed

File Compression and Archiving

tar -cf *archive.tar files* : create an archive called *archive.tar* containing *files*
tar -xf *archive.tar* : extract the archive *archive.tar* and create *files* Additional options:
-j creates or extracts a bz2-compressed version of the archive *archive.tar.bz2*
-r appends files to an existing archive
-z creates or extracts a gzip-compressed version of the archive *archive.tar.gz*
gzip *file* : compresses *file* using gzip compression and names the compressed file *file.gz*
gunzip *file.gz* : uncompresses the file *file.gz*

Useful Keyboard Commands

CTRL-C : halts the current command
CTRL-Z : stops the current command, allowing it to be resumed with **fg** to resume in foreground, or **bg** to resume in background.
CTRL-D : terminates input to a **cat** >*file* command, but if entered at the system prompt, logs out of the current session.
[up-arrow] : recalls commands previously typed in that terminal session.
exit : ends a terminal session.

Additional Resources for Linux Commands

http://www.gnu.org/software/bash/manual/bashref.html is a comprehensive guide to the bash shell; a PDF version is also available.

http://ss64.com/bash/ is a webpage that lists a subset of bash commands, focused on things new users would find useful, with each command linked to a page with more information.

http://tldp.org/LDP/abs/html/ is a tutorial on shell scripting from The Linux Documentation Project, for those who want to learn how to take full advantage of the power of shell scripts. This is a more advanced topic, but well worth the effort to learn if you want to exploit the full power of the Linux environment.

https://tldr.sh/ is a community-generated guide to using command bash commands, with examples

Random Useful Stuff

STDIN : "standard input", usually the keyboard

STDOUT : "standard output", usually the screen

> *file* : redirect STDOUT to *file*, creating *file* if it does not exist, or overwriting if it does. A space between the > and the filename is optional.

>>*file* : redirect STDOUT to append to *file*, without overwriting existing contents.

| : a 'pipe' that connects two commands, so that output from one becomes input to the next. For example: '**cat** *file* | **head –n** 1000 | **tail –n** 20' means display *file* and send to the **head** command, which takes only the first 1000 lines and passes them to the **tail** command, which displays only the last 20 lines to the screen. This allows viewing any specific set of lines out of any file, no matter how large.

sed : a 'stream editor', or an editor that works on the flow of information being processed. It can be used to do 'search and replace' operations with **sed -e** 's/*search-string*/*replace-string*/' *input-file* >*output-file*. This is a very powerful way to manipulate the contents of large text files, analogous to a 'global search and replace' in a word processing program.

man *command* : display the 'man page', or information on how to use a command. To get out of the man page display, type **q**, to move forward in the display type **f**, and to move backward type **b**.

File structure in Linux: the system is organized as a hierarchy of directories, beginning with / (the root directory). A complete file 'path' specifies the location of a file beginning from the root directory, and includes each sub-directory along the way. For example, user directories are typically in the home sub-directory of the root directory, written '/home'. If there are four users – *bob*, *carol*, *ted*, and *alice* – and they each have their own directories in the /home directory, and each of them have a sub-directory called *work* and one called *games* in their home directories, then the complete path to Alice's *quest* program in her *games* directory is */home/alice/games/quest*.

Using the complete path is easiest if the file is in a different directory than the current working directory; if the file is in the current working directory, it can simply be referred to by the file name.

One exception to this is executable programs – the system looks for executable programs only in a set of directories specified by the 'search path', which is set as an 'environmental variable'. To see the current setting of the 'search path', use the command **echo $PATH** at a command prompt. The output will be a set of directory names separated by colons, such as /bin:/usr/bin:/sbin:/usr/local/bin. In order for a program to be executable simply by typing the program name (rather than the full path to the executable file) at a command prompt, either the executable file itself or a link to the executable file must be found in one of the directories in the search path.

UNIX 101: (modified from http://www.uic.edu/depts/accc/software/unixgeneral/unix101.html)

Files

Everything in Unix is a "file" -- real files, directories, device drivers, and so forth. This makes it easy to combine files and programs in many different and new ways.

Directories are special files that hold the names of other files or other directories. In this way, the Unix file system looks like a hierarchical tree, similar to DOS. Note that directories are separated by a forward slash (/), not a backslash (\ ) as in DOS or Windows.

File Attributes

file name

A file name is a string of characters, typically without spaces because the system interprets a space as the end of the filename under some conditions. Unix gives special relevance to some characters – for example, a backslash (\) is an escape character that changes the meaning of the following character, and a filename that begins with a period (.) denotes a hidden file that is not included in the output of the **ls** command unless the **–a** switch (meaning 'all files') is used. In general, Linux does not require the use of extensions to denote file types (such as .docx, .xlsx, or .pptx), although some programs do expect specific types of filenames.

full path

The full path of a file specifies all directories below root, such as **/usr/local/bin/foo.bar**.

directory shortcuts

The current directory is **.** (a single period), the parent directory of the current directory is **..**, and the home directory of a particular user is ~*username*. If no username is specified, then ~ refers to the home directory of the user currently logged in. To execute a program called *program* in the current directory (which is not on the search path), enter the command as *./program* – this tells the system where to find the executable file.

file permissions

Each file has an associated owner and group. The owner is a user, and the group is one or more users. The read, write, and execute permissions (which can be set by the owner) can be different for the owner, group, and public (other). Permissions apply to directories as well as files.

Enter **ls -l** to find the permissions on the files and subdirectories in the current directory. This returns lines that look something like the following:

```
drwx------ 2 bobg comp  512 Jun  7 09:49 mydir
-rwxr-xr-x 1 bobg comp  321 May 30 14:36 myscript
```

Each line describes one file. From left to right: the permissions, the number of links, the owner, the group owner, the size in bytes, the date and time of the last modification, and the file's name.

The first character of the permissions tells what kind of "file" it is; **d** for directory, **l** for link, or hyphen (-) for regular file. The remaining nine characters are three triplets. The triplets give the read, write, and execute permissions for that file or directory for, respectively, the file's owner, its group owner, and for the public (other). The **r** (read), **w** (write), and **x** (execute), indicate the presence of read, write and execute permissions; the hyphen (-) indicates their absence. For directories: **r** permission allows you to list the files in the directory, **w** permission allows you to create or remove files from the directory, and **x** permission allows you to **cd** to the directory.

Thus, in this example, both files are owned by *bobg* and have the group *comp* as group owner. *mydir* is a subdirectory in which only bobg can read, write, and execute; and *myscript* is a file in which bobg can read, write, execute, and everyone else, including those in group comp, can read and execute but not write.