Commonly Used Linux Commands

B. Wilkinson   Modification date: March 4, 2015

Note: When typing commands, you only need to start the name of an existing directory and the "tab" key will cause the name to be completed if the first part of the name is unique.

Basic Terminal Command

clear	Clears the text from the terminal. Useful before seeing just what a subsequent does.

Viewing and Navigating Directories

Frequently used commands for viewing/navigating directories.

ls -a	List files in current directory. **-a** says list all files including hidden files (those starting with a period)
pwd	Print the full path of the current directory
cd *directory*	Change user's directory to that specified, or if none specified, to user's home directory
cd ..	Move up one directory
.	Current directory, often used to prefix the path of a file if the current directory is not in the path for an executable, i.e. **./hello**.

Changing Password

Once you have logged on, you may need to change your password as your account may have been set up with a password that was sent to you in an insecure way (say by email). The command is

passwd	Change user password, which will prompt for the current password and then the new password.

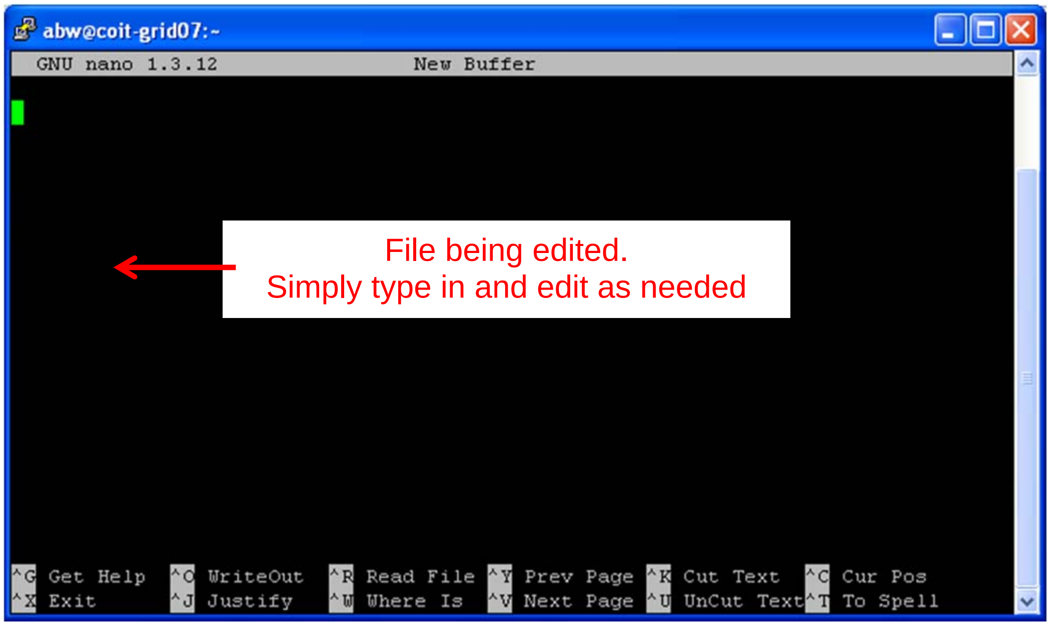
Creating Directories and Files

Program files are usually created using an editor.

Basic Linux Editor: The standard Linux editor is called **vi**. One simple editor that may be more convenient for the casual user and part of most Linux installations is called **nano**. To invoke **nano**, type on the command line:

nano *filename*

or without a file name if file not yet created. One simply types into the window to change the contents the file. Use control-O to save and control-X to exit.   The available commands are given at the bottom of the window:


```
abw@coit-grid07:~
  GNU nano 1.3.12                New Buffer

^G Get Help   ^O WriteOut   ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```

Control key commands: **Control-O** to save, **Control-X** to exit

Ubuntu: A very convenient graphical editor is available on Ubuntu called **gedit**, see separate notes on Ubuntu for more information on this editor It is not available on the UNCC cluster.

Manipulating Directories and Files

Common commands for manipulating directories and files:

mkdir *directories*	Create one or more directories
cat *files*	Displays contents of files.
more *file*	
less *file*	Both display file one page at a time (space bar to go forward one page, 'b' to go backward one page, 'q' to quit)
cp *file1 file2*	Copy *file1* to *file2*
cp *file1 directory*	Copy *file1* to *directory*
mv *sources destination*	Move or rename files and directories
rm *files*	Delete one of more files
rmdir *directories*	Delete one of more directories. Must be empty.
rmdir -r *directories*	Remove non-empty directories (recursive remove)

Getting information, killing jobs and logging out

man *command*	List information about command, arguments etc.

	Type **q** to leave before end.
ps	Display information about processes.
kill *n*	Kills process *n* where *n* is the process number, which can be obtained from **ps** command
exit	Terminate your login shell

Secure file copy to/from remote servers on command line

scp [*user@*]*host1***:***file1*] [*user@*]*host2***:***file2* Copies files between hosts on a network

Example

To copy the file **matrixmult.c** from the local machine to **cci-gridgw** and place in an existing directory **/ParallelProg/OpenMP**, issue the command on the local machine:

scp matrixmult.c <username>@cci-gridgw.uncc.edu:/ParallelProg/OpenMP

where **<username>** is your username on **cci-gridgw.uncc.edu**. You will be prompted for your password on **cci-gridgw.uncc.edu**.

Using C compiler

The regularly installed default Linux C compiler is called **gcc** invoked with the command:

cc <options> filename

The usual option is to specify a name for the executable using **–o** (output) option either before or after the source file name, for example:

cc –o hello hello.c

would compile **hello.c** and create the executable called **hello**. Without the **–o** option, the name of the executable is **a.out**. *Some prefer putting the* ***–o*** *option at the end to avoid accidently deleting the source file if the executable name is omitted!*

Note the current directory is usually not in the path to execute an executable, i.e. so to execute use: **./hello**.