

Linux terminal commands

Linux has a very powerful command-line interface, which is invoked by typing commands into a terminal or xterm window directly (like the DOS/CMD window in Windows). This small note can help you get started learning some of these commands; remember that much more detailed descriptions are available online, in particular:

<http://www.redhat.com/docs/manuals/linux/RHL-6.2-Manual/getting-started-guide/ch-doslinux.html>
<http://www.hscripts.com/tutorials/linux-commands/>

In the labs, click on **Applications->accessories->terminal** to open up the terminal window which gives you access to the command prompt. If you log into the machines from home using PuTTY, then you type these commands in the PuTTY window.

Here are some common commands:

- **mkdir**: *make directory* (a directory is often called a *folder* in windows). A directory is a special file on the storage device that can contain other files. For instance, `mkdir cs1073` will create the directory/folder `cs1073` in the present working directory.
- **pwd**: *print working directory*. Shows where you are in the file system (i.e. are you at your top directory, or deep in nested folders)? For instance, right after I log in, if I type `pwd`, I get:

```
main205>pwd
/home/whaley
```

- **cd**: *change directory*. This is the way to move around in the filesystem. For instance, if I type `cd cs1073` then I will change into the subdirectory (a subdirectory is a folder which is contained within another folder) `cs1073`.

⇒ There are two special directory handles that you use when issuing command like `cd`. These are:

1. `./`: the current directory the command prompt is in. For example `./xc2f` runs the program `xc2f` which can be found in the present working directory.
2. `../`: the directory immediately above the current one. Can be repeated as often as you like. For instance `../xc2f` runs the program `xc2f` which is found in the directory two levels above the current one.

⇒ When you provide where the file is to be found, this is called the *path*. So for `../xc2f`, `../..` is the *relative path* (because the path is relative to the current working directory), and `xc2f` is the *file name*. A *fully qualified path* or *absolute path* is one that is not dependent on the directory where you currently are. For instance `/home/whaley/cs1073` is a fully qualified path.

- **ls**: *list*. Prints out what files are available in the present working directory. Important flags include `-A`, which says to list hidden files as well as normal files, and `-l` which says to list with long format (providing more information about all files).
- **man**: *manual*. Print man page of provided command. For instance, `man ls` provides the system help on using the command `ls`.

- **cp**: *copy*. Copy a file to another file. For instance ‘`cp c2f.f90 f2c.f90`’ will copy the file `c2f.f90` so that the same data exists as the file `f2c.f90` in the same directory.
 - Linux filename commands have the special character ‘.’, which means “Using the same filename” when given as a target for commands like `cp` or `mv`. For instance ‘`cp c2f.f90 ../.`’ will duplicate the file `c2f.f90` into the directory above this one, using the filename `c2f.f90`.
- **mv**: *move*. Copy a file to a new file/path, and delete the original file. For instance, ‘`mv c2f.f90 celc2fair.f90`’ will result in replicating the file `c2f.f90`, and then deleting the original filename (`c2f.f90`).
- **rm**: *remove*. Delete the provided file(s). For instance ‘`rm c2f.o`’ will permanently delete the file `c2f.o`. **Use this command with care**, since you can delete every file you own if you are not careful. I recommend that you always use the `-i` flag, which will cause `rm` to ask you if you really want to delete. You can ensure this by typing ‘`alias rm rm -i`’ before ever using `rm`.
- **rmdir**: *remove directory*. Delete the specified directory. This command only works if the directory is empty.
- **history**: print a listing of most recent commands the user has entered. You can then repeat a command using the `!`. For instance `!110` will repeat the command with the label 110 from the `history` listing.
 - For most shells, hitting the up arrow will take you one command back in your history, the down arrow will take you one command forward in your history, and the left and right arrows allow you to move around in these remembered commands so that you can edit them.
- **cat** : *concatenate*. Takes a list of text files, and prints them to screen, starting with the first file in the list and ending with the last. If used with only one file, it types its contents to the screen.
- **diff**: *difference*. Compares two different text files, and displays any differences line-by-line to the screen. If there is no output, then the files contain the exact same text. Very useful for making your output match someone else’s exactly, or to see what has changed between versions of a program.
- **grep**: *global regular expression print*: search for given pattern/patterns in the provide list of files and print any occurrences line-by-line to the screen. Has simple form `grep pattern [files]`.
- **gfortran** : invoke the Fortran 95 compiler. For example: ‘`gfortran -g -Wall -o xc2f celc2fair.f90`’ will compile the program that has been saved to the filename `celc2fair.f90` in the present working directory, and place the executable in the filename `xc2f` in the same directory (under windows, a `.exe` file extension will be automatically added to the filename, but this is not true for linux).
- **mail yhan@cs.utsa.edu < program.f90**: mail the file `program.f90` to the TA. You should replace `program.f90` with the filename of your program. You should have comments describing your name and any collaborators at the top of the file.
- **pine**: a simple non-graphical e-mail client

Example session

The following is an actual screen dump of a terminal session showing how one can use some of the outlined commands (I have added some blank lines to make it slightly easier to follow). I start on my office machine (`drteeth`), and I log into the machine `main201.cs.utsa.edu` (I do this using `ssh` since my machine runs linux; OS X users can also use `ssh`; Windows users can use PuTTY). In this session, I create a directory for all of my CS 1073 files, and then I create subdirectories so that it is easy to find files later. Note that the sentences prefaced with `#` were not typed in by me or printed out by the terminal: they are comments that I have added to explain what I was doing during the session.

The *prompt* is where you type the commands in the terminal window. Different users have different prompts (and you can change them if you know how). My prompt is `machine_name>`. So, on `main201` it will be `'main201>'`. Your prompt may differ.

The machines `main201` through `main205` should be available for remote login to linux 24 hours a day. Other machines are available or not depending on if they have been booted into Linux or Windows. These machines include `ant00 – ant34`, `bat00 – bat55`, `cat00 – cat32`, `dog00 – dog62`.

Remember that if you type the first few letters of a file name and then hit the Tab key, the shell will usually autocomplete the file name!

```
drteeth>ssh main201
whaley@main201's password:
Last login: Tue Jul 21 15:47:13 2009 from drteeth.cs.utsa.edu
main201>pwd
/home/whaley
main201>mkdir cs1073                # create class dir in home area
main201>cd cs1073/                  # go to class subdir
main201>ls                          # no files in subdir yet
main201>ls -a                       # -a shows hidden/virtual files
./  ../
main201>cp ../lec/* .               # copy all files in ../lec
main201>ls                          # see what files I copied in
basics.pdf  centigrade.f90  intro.pdf  loser.f90  xconvert*
c2f.exe*   centigrade.f90~  labinst.pdf  syll.pdf  xpoke*

main201>rm *.exe x* *.pdf centigrade.f90~  # get rid of unneeded files
main201>ls
centigrade.f90  loser.f90
main201>mv centigrade.f90 cent2faih.f90    # get better name for program
main201>ls
cent2faih.f90  loser.f90

main201>gfortran -Wall -g -o xc2f cent2faih.f90 # compile program
main201>ls                                       # see if executable is there
cent2faih.f90  loser.f90  xc2f*
```

```

main201>./xc2f                                     # run program
  What is the temperature in Centigrade?
100
  100.0000      degrees Centigrade is      212.0000      degrees in Fahrenheit

main201>gfortran -Wall -g -o xloser loser.f90      # compile program 2
main201>./xloser                                   # run prog 2
  Please enter an integer between 1 and 500
2
  Please enter your first name:
Clint
  Thanks Clint! Now please enter your last name:
Whaley
  Clint Whaley is a complete loser!
  Clint Whaley is a complete loser!

main201>ls
cent2faih.f90  loser.f90  xc2f*  xloser*
main201>rm x*                                     # get rid of executables
main201>ls                                         # see files that remain
cent2faih.f90  loser.f90
main201>mkdir lab1                                # create directory for 1st lab
main201>mv *.f90 lab1/.                            # put f90 files into it
main201>ls
lab1/
main201>ls lab1/                                  # make sure they are there
cent2faih.f90  loser.f90

main201>mkdir asg1                                # create dir for asg 1
main201>cp lab1/cent2faih.f90 asg1/faih2cent.f90# use c2f as start for f2c
main201>ls asg1
faih2cent.f90
main201>cd asg1 ; ls                              # cd to asg1, list files
faih2cent.f90

```

Additional Command-line Information

- **File completion:** when typing, the shell can autocomplete most words for you. Type the first several characters of the word, and then hit the `tab` key. If the filename is not autocompleted, there may be other filenames that begin with the same prefix, so you need to type a few more characters and hit `tab` again until the prefix you have typed is unambiguous. In the bash shell, hitting `tab` twice will give a list of possible completions, while `ctrl-D` does the same for the tcsh shell.
- **Unix wild-carding:** On the command line the `*` character means "match anything", while `?` means match any one character, while the brackets provide a way to match a range or list of certain characters. Here's some examples:
 - `ls *.dat`: lists all files in current directory ending in `.dat`.
 - `ls *hello*` : lists all files with the word `hello` embedded in their names
 - `cat output[0-9].out`: prints out all files with the form `output` followed by a single digit, followed by `.out` to the screen.
 - `cat output[G,B].out`: prints the contents of the file `outputB.out` followed by the contents of `outputG.out` to the screen.
 - `ls hello?.f90` : lists all filenames that consist of `hello`, followed by any character, followed by `.f90`.
- **Redirection and pipes.** Unix has a facility allowing you to redirect input and output of programs into files, and to connect the output of one program to another using a pipe:
 - **Output redirection:** The `>` placed after a program/command tells unix to redirect the standard output of a program or command (which usually goes to the screen) to another place. It is usually used to redirect output to a file. Therefore, the command:

```
cat bob1.out bob2.out > bob_combined.out
```

will create a new file which has the contents of `bob1.out` in its first lines, followed by the contents of `bob2.out`.
 - **Input redirection:** The command `<` redirects the contents of a file (usually) to the standard input of a program. Therefore, the command:

```
/xprogram < test.in
```

will use the contents of `test.in` as input to the program `xprogram`. It will be pretty much like you ran `xprogram`, and then manually typed the contents of `test.in` manually in response to the input prompts from the program.
 - **Piping:** you can pipe the standard out of one command/program directly to another command/program using the pipe symbol, `|`. This allows you to hook up a series of filters that all take the output of some program, modify it, pass it on for as many filters as you like. For example:
 - * `cat test.in | ./xprogram.in` is the same as the input redirection given above.
 - * `ls -l *.out | grep -i joe`: This will cause `ls` to generate a list of all

filenames ending in `.out`, which will be passed as input to the `grep` command, which will then print any such file that also contains the word `joe` embedded in it (irrespective of case).