Quick Intro to Linux

CS - UTSA

Terminal Commands

Linux has a very powerful command-line interface, which is invoked by typing commands into a terminal or xterm window directly. This guide will help you begin to navigate the Linux terminal and construct simple C programs. One of the skills of a computer programmer is to search for solutions to the problems we are trying to solve. Below are suggested YouTube videos. It is recommended that you get in the habit of searching out additional explanations of computer science topics that we will be studying.

```
Linux Command-Line Tutorial - http://bit.ly/LinComTut
C Programming Tutorial - http://bit.ly/CProgTut
```

To get a jump start on our studies, we will be using an online emulator to practice Linux commands and simple C programs. If you have access to a Linux or Mac computer, you can practice on your computer. Windows users should practice on the website until you install the proper software.

```
Linux Command-Line Emulator - http://bit.ly/LinuxEm
```

Each line will have the name of the computer or server followed by a cursor. Once your semester has officially started, you will have access to the UTSA fox servers with a command line like this:

```
fox03:>
```

And each line on the emulator looks like this:

```
[root@localhost ~]#
```

Here are some common commands:

- `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.
- `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 logging in, type '`pwd`' to get:

  ```
  fox03:> pwd
  /home/korkmaz
  ```

- `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 cs2213`' will create the directory/folder `cs2213` in the present working directory.

- `cd` : *change directory*. This is the way to move around in the filesystem. For instance, if I type `cd cs2213` then I will change into the subdirectory (a subdirectory is a folder which is contained within another folder) `cs2213`. To move up one directory level, type `cd ..`

- `vi filename` : *an editor*. Take this opportunity to search for vi tutorials. vi is easy to use but different from a regular text editor.

- `cat` : *prints*. Prints out the content of files on the screen.

- `cp`: *copy*. Copy a file to another file. For instance '`cp c2f.c f2c.c`' will copy the file `c2f.c` so that the same data exists as the file `f2c.c` 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.c ../.`' will duplicate the file `c2f.c` into the directory above this one, using the filename `c2f.c`.

- `mv`: *move*. Copy a file to a new file/path, and delete the original file. For instance, '`mv c2f.c celc2fair.c`' will result in replicating the file `c2f.c` as `celc2fair.c`, and then deleting the original filename (`c2f.c`).

- `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.

- `man` : *manual*. Print man page of provided command. For instance, '`man ls`' provides the system help on using the command `ls`.

- `gcc` : invoke the C compiler. (more information is in the next page)

Creating Files & Compiling C Programs

- To create a file with the name of `program.c`, we need a text editor:

  1. Text editors that work within a terminal window include vi, vi, vim, and emacs. To use these simply issue a command like: `vi program.c` in the terminal window. This will create a new program.c file if none exists already. It will open program.c if it already exists.
     Editing with `vi`: Navigate to the directory you want to work in. Open vi with a given file name. Remember, type 'i' for insert mode and `esc :wq` to save and quit.
  2. You can also use a graphical text editor similar to Notepad (Sublime Text is a favorite). Make sure your terminal is in the same directory that you are saving the file. Save in the graphical editor then compile on the terminal.
  3. vi and the gcc compiler will work in the emulator but will not save your work between sessions.

- Once you are in a text editor, you can type any text and save it. Here is a famous C program that is worth typing and saving:

```
#include <stdio.h>

int main()
{
   printf("Hello, world.\n");

   return 0;
}
```

- To compile the program that has been saved to the filename `program.c` in the present working directory and create an executable:

```
fox03:> gcc program.c -o program
or
fox03:> gcc -o program program.c
```

  (where of course `program` is replaced by the name of your new program).

- To execute the program, simply type `./program`.

Make a directory hierarchy for this class

1. If the semester has started, log into one of the UTSA servers, **fox01**. . . **fox06**, otherwise use your own machine or the emulator.

2. Make the directory `courses` under your home directory with the command `mkdir courses`

3. Change directories to your `courses` directory with the command `cd courses`
4. Make the directory `cs` under your `courses` directory.
5. Change directories to your `cs` directory.
6. Make the directory `2213` under your `cs` directory.
7. Change directories to your `2213` directory.
8. Change to your home directory using `cd`. Notice that a `cd` with no argument changes to your home directory.
9. Print the working directory using the command `pwd`.
10. Practice creating directories for your other classes.

A sample 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).

In this session, as defined above, I create a directory hierarchy for all of my cs 2213 files. Subdirectories will allow us to easily 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.

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!

```
fox03:> pwd                                     # shows current directory
/home/korkmaz
fox03:> mkdir courses                           # create courses dir in home area
fox03:> cd courses                              # go to courses subdir
fox03:> ls                                      # no files in subdir yet
fox03:> ls -a                                   # -a shows hidden/virtual files
./  ../
fox03:> pwd                                     # shows current directory
/home/korkmaz/courses
fox03:> mkdir cs                                # create cs dir in courses
fox03:> cd cs                                   # go to cs subdir under courses
fox03:> mkdir 2213; cd 2213

fox03:> vi program.c     #gedit program.c &   # create a file, edit/save as above
fox03:> cp program.c prog.c                     # copy program.c to prog.c

fox03:> gcc -o prog prog.c                # compile prog.c
fox03:> ls                                      # see what files I have now
program.c prog.c prog
```

```
fox03:> rm *.exe x* prog prog.c~               # get rid of unneeded files
fox03:> ls
program.c prog.c prog

fox03:> mv prog.c helloworld.c                # get better name for program
fox03:> ls
program.c helloworld.c

fox03:> gcc -g -o hello helloworld.c      # compile program: -g for debug

fox03:> mkdir ex1                             # create directory for 1st example
fox03:> mv *.c ex1/.                          # put c files into it
fox03:> ls
hello ex1/
fox03:> ls ex1/                               # make sure they are there
program.c helloworld.c
```

Let's create an another program under `ex1` to convert Centigrade to Fahrenheit:

```
fox03:> cd ex1                            # go to ex1 directory
fox03:> mv program.c c2f.c                # move program.c to a new file named c2f.c
fox03:> vi c2f.c    # gedit c2f.c &     # open the file with an editor
```

In the editor, modify the content so that you will have the followings

```
/*
 * Include C header files
 */
#include <stdio.h>
#include <stdlib.h>

int main(int nargs, char **args)
{
   double temp_c,   /* temperature in Centigrade */
          temp_f;   /* temperature in Fahrenheit */
   /*
    * Prompt for centigrade temperature
    */
   printf("What is the temperature in Centigrade? ");
   scanf("%lf", &temp_c);
   /*
    * Convert it to Fahrenheit
    */
   temp_f = 9.0*temp_c/5.0 + 32.0;
   printf("%.2f degrees Centigrade is %.2f degrees in Fahrenheit\n",
          temp_c, temp_f);
   return 0;        /* signal normal end of program */
}
```

Now let's compile and run it.

```
fox03:> gcc -g -o c2f c2f.c            # compile program: -g for debug

fox03:> ls                                   # see if executable is there
c2f.c  c2f* helloworld.c

fox03:> ./c2f                                # run program
 What is the temperature in Centigrade? 100
   100.0000     degrees Centigrade is    212.0000     degrees in Fahrenheit
```