System Administration for Beginners

Supplementary #1: Getting Around in UNIX

February 16, 2007

I have written this guide up because I feel like lecture is focusing a bit too much on *telling* you how things work or how they should work rather than *showing* you. Included in this document are some of the basics that are key to mastering before moving further into more advanced topics in.

This supplementary guide will focus on the topics of learning the commands and what they do rather than explain concepts. Though important, they have been covered in the lecture notes. Some words my be *emphasized* like so to point out concepts or terminology that would be a good idea to look up.

If you're just starting, don't worry too much about memorizing commands or let the acronyms get to you. After some practice (like working on lab exercises and homework), the commands and steps will come to you easily. Otherwise, make use of the `man` pages; if you are ever unsure of what a command does, be sure to read up on the manual before executing it.

Also, as a word of warning, UNIX and other UNIX clones will not complain much unless a command is entered incorrectly. If you enter the command `rm -rf /`, which will delete the entire root directory (effectively killing your system), the operating system will assume you know what you are doing and does not hesitate to execute that operation.

1 Getting Started

1.1 Logging In

There are two basic ways to log in to a computer and pull up a *terminal*. The first way is if you are physically present at the machine with a keyboard, monitor, and mouse. There will be a prompt waiting for you to enter your user name, then prompt you for your password, and then proceed to log you in. A *GUI* will generally be provided; much like the equivalent of Themes on Microsoft Windows, the GUI, or window manager, can be changed and customized as you see fit. Though this topic will not be covered here, you can find more information on it online. [3]

The second way is to remotely connect to it through *SSH*. Depending on your operating system, there are different terminals that you can use to connect to the computer. For Microsoft Windows, popular ones include SSH Secure

Shell [2] or PuTTY [1] and will generally prompt you for the host to connect to and your login. On Mac OS X and GNU/Linux, a terminal should have been installed by default and you will typically use the command line to connect to a different server.

For the purposes of this course, you will log in to a computer in a computer lab or your computer at home and remotely connect to another computer (i.e., your project server) located at the Open Computing Facility (OCF). Though the machines are physically present somewhere on campus, you will only have remote access to them.

1.2 Working on the Command-Line

After logging in, you will want to open a terminal (like `xterm` or `eterm`). You should be greeted with something like:

```
[cardi@fallingrocks]~$
```

The terminal that you opened is running a *shell*, this is where you will input commands and where display will generally output. If you type something in, it will appear directly after the $. The syntax for the information preceding the $ is as follows: [user-name@machine-name]directory$. Though it may seem redundant, this is very useful for knowing what login you are using and which machine you are remotely connected to. The syntax for the above differs from shell to shell, but can generally be customized by editing a configuration file.

Now, we want to SSH into a computer which we want to manage. For the purposes of this document, we will be using the services of the OCF; most of the commands here should be reproducible on other systems.

TIP I will denote when to press the enter key with ↩.

Using the `ssh` command, we will remotely connect to one of the OCF's servers. Note the command's syntax. If the connection is successful, it will prompt me for the password and then I will be a given a shell to execute more commands on.

```
[cardi@fallingrocks]~$ ssh cardi@ocf.berkeley.edu ↩
Password:  (type your password here) ↩

[cardi@apocalypse]~$
```

When finished using the terminal, make sure to exit using the command `exit`. Depending on the terminal you are using, it will either close or grey out the window. Security wise, it is important that you close any terminals after you are finished, especially ones with `root` logged into them. Otherwise, someone could easily hijack your terminal and perform malcious commands under a privileged user.

```
[cardi@fallingrocks]~$ exit ↩
```

2 Basic Commands

2.1 Traversing Directories

Now that we have logged in, we can start using (and soon, administrating) the system we are connected to. Let's start by seeing where we are in the filesystem. To print the working directory, use the command `pwd`.

```
[cardi@apocalypse]~$ pwd ↩
/home/c/ca/cardi
[cardi@apocalypse]~$
```

Once I input the command, I hit the enter key (↩). The shell interpreted this and behind the scenes, sent instructions to the operating system to interpret your command. What results is an output that we can understand and then another prompt for the next command you want to enter.d behind the scenes, sent instructions to the operating system to interpret your command. What results is an output that we can understand and then another prompt for the next command you want to enter.

The result of the command is the printout of the directory we are currently in. To imagine the directory structure, think of whatever is in between the two "/" as an actual labeled manila folder. In this example, the folder "`cardi`" is a subdirectory, or inside, the folder named "`ca`", which in turn is a subdirectory of "`c`", and so on, until the root directory `/`.

All our work will not be done in one directory, however. At the moment, we are currently in our home directory (in shorthand: ~). This is where we will store our personal files. To change directories, we will use the command `cd` ***`path`***. *Relative* and *absolute* paths will not be covered here, but it is important to differentiate the two.

Here is an example of traversing directories. Note the change in the text preceding the command prompt.

```
[cardi@apocalypse]~$ cd ..  ↩
[cardi@apocalypse]/home/c/ca$ pwd ↩
/home/c/ca
[cardi@apocalypse]/home/c/ca$ cd ..  ↩
[cardi@apocalypse]/home/c$ cd ../..  ↩
[cardi@apocalypse]/$ pwd ↩
/
[cardi@apocalypse]/$ cd ~ ↩
[cardi@apocalypse]~$ pwd ↩
/home/c/ca/cardi
[cardi@apocalypse]~$
```

2.2 Working with and Manipulating Files

2.2.1 Listing Contents

It is rather easy to move around directories; there are so many commands that you need to know in order to get where you want to go. However, how do I know which directories to `cd` to? One way would be to simply memorize and keep track of all the directories that are existing, but after a while, that could probably become very tedious. Fortunately, there are more commands that we will learn that will aid us in this situation.

NOTE From this point forward, I will try to only show the relevant output. Generally after entering a command and displaying the output, the shell will again create a new line waiting for command input.

To list the contents of a directory, we can use the command `ls`. Using this command will list the files and other directories that we may have in our current working directory.

```
[cardi@apocalypse]~$ ls ↩
Deskop Documents
```

We now see the contents of the current directory. UNIX does not differentiate between files and directories in terms of naming. Some shells may assign a different color to show which are files and which are directories. To figure out what something may be, we will use the `file` command.

```
[cardi@apocalypse]~$ file Documents ↩
Documents:    directory
```

We can even input two or more filenames and it will output what each file type is.

```
[cardi@apocalypse]~$ file Documents Desktop ↩
Documents:    directory
Desktop:    directory
```

Knowing that they are directories, we can `cd` into them and work with whatever we need to from there. `ls` does not show all the files in a directory, however. Files that have a "`.`" as the first character of the filename are hidden. In order to see them, you will need to pass a flag; in this case, `-a` will print out all the files in the directory.

```
[cardi@apocalypse]~$ ls -a ↩

.              .dt              .gtkrc-1.2-gnome2  .softwareupdate
..             .dtprofile       .icons             .ssh
.ICEauthority  .esd_auth        .login             .sunw
.TTauthority   .gconf           .macromedia        Desktop
.Xauthority    .gconfd          .metacity          Documents
.adobe         .gnome           .mozilla
.bash_history  .gnome2          .nautilus
.cshrc         .gnome2_private  .recently-used
```

There are a lot more files than shown before. Many of the files listed are "hidden" because they contain important configuration information that makes things work, like your shell or the GUI that you are using. They are editable, but usually you will only have to edit them once or twice to get it the way you want and leave it alone.

Try entering the command `ls -al`. What more information does this give you? If you're unsure, look at lecture notes from week 3 to understand the concepts of *ownership* and *permissions*.

2.2.2 File Manipulation

The directory we're working with looks pretty empty. Let's start creating, modifying, and deleting files. To create an empty named file, we can use the command `touch`. The file will be of size 0 and considered "empty".

```
[cardi@apocalypse]~$ touch work ↩
[cardi@apocalypse]~$ ls ↩
Desktop Documents work
[cardi@apocalypse]~$ file work ↩
work:    empty file
[cardi@apocalypse]~$ du work ↩
0      work
[cardi@apocalypse]~$ rm work ↩
[cardi@apocalypse]~$ ls ↩
Desktop Documents
```

In summary, I created an empty file called "work", listed the contents of the directory (`ls`), tried to figure out the type and size of the file (`file` and `du`), and then deleted it (`rm`).

What are the equivalents to creating and deleting directories? Try looking at lecture notes from week 2 for a listing of some of the commands.

References

[1] http://www.chiark.greenend.org.uk/ sgtatham/putty/.

[2] http://www.ssh.com.

[3] http://xwinman.org/intro.php.