Introduction to Unix

May 10, 2009

Exercises: Using Commands

1. **Create a new user:**

```
# pw useradd inst -G wheel -m -s /usr/local/bin/bash
```

Use "`man pw`" to understand what each option above does.

Set the password for the new user "inst" to be "`success2k8`":

```
# passwd inst

Changing local password for inst
New Password:  <success2k8>
Retype New Password: <success2k8>
```

Log in as the new user you just created:

```
# logout

login: inst
password: <success2k89>

[inst@pcN ~]$
```

When we use your *inst* account we'll use a "$" to represent the command prompt. When we want you to use the *root* account we'll use a "#" to represent the command prompt.

2. **View files:**

Use `ls` to list files:

```
$ cd                                    [go to your home directory]
$ ls
```

Do you see anything? Try this instead:

```
$ ls -lah
```

What's inside one of these files?

```
$ cat .cshrc
```

Try:

```
$ less .cshrc
```

Press ctrl-c to get out of the `less` display.

Now try:

```
$ clear
$ cat .cshrc
```

If you don't understand what `cat, clear` or `less` did, then type:

```
$ man cat
$ man clear
$ man less
```

And, now try:

```
$ more .cshrc
```

3. **Working with the command prompt:**

   You can recover previous commands by using the up-arrow and down-arrow keys. Give this a try now.

   Alternately, try typing this command:

   ```
   $ history
   ```

   If you wish to execute one of the commands in the list you saw type:

   ```
   $!nn
   ```

   Where “nn” is the number of the command in the history list. This is useful if you want to run a past command that was long and/or complicated.

   Command completion:

   With the bash shell you can auto-complete commands using the tab key. This means, if you type part of a command, once you have a unique string if you press the TAB key the command will complete. If you press the TAB key twice you'll see all your available options. Your instructor will demonstrate this, but give it a try by doing:

   ```
   $ hist <TAB>
   $ del <TAB><TAB>
   $ rm <TAB><TAB>          [Include the space after the “rm”]
   ```

4. **Working with pipes:**

   We saw an example of using pipes when we sorted the contents of our /sbin directory during the presentation. What if you wanted to have this information available in a file and sorted?

   ```
   $ cd
   $ ls /sbin | sort > sbin.txt
   ```

Now view the contents of what is in sbin.txt to verify that this worked. Do you remember how to do this? If not, go back to exercise 2.

5. **Finding text strings:**

   Use the command `grep` to print lines matching a pattern in a data stream (such as a file). For example, view the entry for the *inst* account in the system passwd file:

   ```
   $ grep inst /etc/passwd
   ```

   You should see something like

   ```
   inst:*:1001:1001::/home/inst:/usr/local/bin/bash
   ```

   The items above are:

   ```
   userid:passwd:uid:gid:Name:HomeDir:LoginShell
   ```

   `grep` is often used with a pipe to reduce the number of results. For instance:

   ```
   $ history | grep ls
   ```

   Will display your previous use of the `ls` command from exercise 2.

6. **Editing the command line revisited:**

   It is particularly useful to realize that you can edit a command just as you would a line of text in a file. For instance, you can:

   - Use your back-arrow and forward-arrow keys to change text in a command.
   - Use the Home and End keys to go to the start and the end of a command.
   - Note: you *do not* need to go to the end of a command before pressing <ENTER> to execute the command.
   - You can use the `history` command with `grep` to find a previous command. You can copy and paste this command, then edit it to make adjustments. For long commands this can save considerable time.
   - Alternatively you can use the reverse-search feature of bash:
     1.) Press ctrl-r.
     2.) type the term you are searching for.
     3.) Press ctrl-r to cycle through all occurrences of the term in your history.
     4.) Press the right or left-arrow, HOME or END key to start editing the command.

   Let's give some of these editing rules a try:

   ```
   $ ls -lah /usr/local/bin | grep perl*
   ```

   Then, let's look for perl in /usr/local/lib. Try this:

   ctrl-r, type “perl”, then press left arrow. Edit the previous command (which you should now have) and change “bin” to “lib”. Use the left-arrow key to move. You should now have:

   ```
   $ ls -lah /usr/local/lib | grep perl*
   ```

With your cursor just past the “b” in “lib”. Press <ENTER> to execute the command.

7. **Copy and pasting commands:**

   A nice feature in Unix is the built-in automatic copy buffer (think “automatic clipboard”). As soon as you highlight text it is available in your copy buffer. You *do not* need need to use the ctrl-c, ctrl-v keyboard combination to copy and paste text.

   After you have highlighted text, then you can place your cursor where you want to paste the highlighted text and press the *middle* mouse button to do the text paste.

   Give this a try:

   ```
   $ history | grep perl
   ```

   Locate the previous command from exercise 5 and highlight just that command. Highlight the command text (not the number next to it).

   Now press the middle mouse key. The command should be pasted to your prompt.

   Did this work? Now you could edit the command or just press enter to execute it.

   This is very useful if you have two terminals open and want to use text from one terminal in a different terminal. Or, if you wish to copy resulting text from a command in one terminal in to a file that's open in another terminal – etc.

8. **Virtual terminals:**

   We will do this exercise together in class.

   We will show you how to use multiple terminals in Unix when you *do not* have a Graphical Interface available to you, and how to copy and paste between the virtual terminals.

   This can save time when administering a Unix server from a console.