Fun with Linux

From Education

Creating and Cat-ing

Now that you've heard the Introduction to Linux presentation, it's time to practice some basic Linux commands. First, remember the rule of thumb - know thyself! If you ever forget, just type `whoami`. (Try this now.) It's also helpful to know where you are. `pwd` stands for print working directory, and it will show you.

Let's make a folder to play around in: do this with the command `mkdir <directoryname>`. Once you've done that, use `ls` to see that your directory has been created, then move into it with `cd <directoryname>`.

```
kwanous@eyrie:~$ mkdir funwithlinux
kwanous@eyrie:~$ ls
funwithlinux  hosts
kwanous@eyrie:~$ cd funwithlinux/
```

Let's create a file to play around with. To create an empty file in Linux, use the command `touch <filename>`. For example,

```
kwanous@eyrie:~/funwithlinux$ touch mynewfile
kwanous@eyrie:~/funwithlinux$ ls
mynewfile
```

Right now this file has nothing in it. We can verify this with the `cat` command. `Cat`, short for concatenate, prints the contents of a file to the screen. If you type `cat <filename>`, nothing should be printed. Let's add something to the file. To open the file in a text editor called "nano", type `nano <filename>`. Go ahead and type a few different lines in your file, something like the following, and then save it.

```
Hi!  My name is Kristina.
I am having fun with Linux!
This is in my file called mynewfile.
```

Then use `cat <filename>` again. The new contents of the file should be displayed.

Copying and Moving

Let's say we want to change this file's name. The way we do this is by moving it to a new location. The syntax for this is `mv <oldname> <newname>`. For instance,

```
kwanous@eyrie:~/funwithlinux$ mv mynewfile MyNewFile
kwanous@eyrie:~/funwithlinux$ ls
MyNewFile
```

Notice that the original copy is gone. What if we wanted to keep the original and make changes to the new one? To make a copy of a file, use `cp <original> <newfilename>`. Go ahead and make mynewfile again using `cp`.

Use `ls` to make sure both `mynewfile` and `MyNewFile` are in your directory.

Let's change `mynewfile` to make sure that it and `MyNewFile` are really two separate files. Use nano to change `mynewfile` and save it. Then `cat` both of your files. (Is Linux case sensitive? How do you know?)

We can also move files to a different directory. Create a new directory called firstexample and then move the files into this directory using either an absolute or relative location.

Relative positioning:

```
kwanous@eyrie:~/funwithlinux$ mv mynewfile firstexample/
```

Absolute positioning:

```
kwanous@eyrie:~/funwithlinux$ mv mynewfile ~/kwanous/funwithlinux/firstexample/
```

Ls and Wildcards

Now, let's create more files to play around with. Use `touch` to create the files pickel, pear, peach, apple, and prune. Rather than typing these in one line at a time, you can put them all on one line like this:

```
kwanous@eyrie:~/funwithlinux$ touch pickel pear peach apple prune
```

Oops! I spelled pickle wrong. If you followed my example and did too, use `mv` to change the file name.

Typing `ls` will show all of the new files you've created. To see more information about the files in a list format, try `ls -l`. This shows all of the files in the directory you're currently in. But what if we only wanted the files that started with p? Try `ls p*` or `ls -l p*`. Is Linux case sensitive? Try `ls P*` to find out.

The * is called a wildcard character. What exactly does it mean? Try a few experiments to find out (like `ls peach*` and `ls *e` and ls *). How would you display the files that start with p and end with e?

Removing Files and Directories

Let's moving on to removing items. Let's remove peach. Use `rm peach`, then do an `ls` to make sure it's really gone. But what if we wanted to remove multiple files? The wildcard character works the same here as it does with ls! (Be very careful with `rm *`!) Go ahead and remove apple and grape with one command.

Next let's remove the directory. Try `rm <directoryname>`.

```
kwanous@eyrie:~/funwithlinux$ rm firstexample/
rm: cannot remove `firstexample/': Is a directory
```

What happened? `rm` is letting you know that this is a directory. Just like there's a special command to create directories (`mkdir`), there's a special command to remove directories so that people don't accidentally remove a directory thinking that it's a file. It's used as `rmdir <directoryname>`. Hm... if you had files in your folder like I did, it still won't work. Rmdir gives us the reason why, too.

```
kwanous@eyrie:~/funwithlinux$ rmdir firstexample/
rmdir: firstexample/: Directory not empty
```

This is a protection mechanism built in to help users not accidentally delete folders with files still in them. One way we could remove everything would be to move into the directory and then remove them. However, we can also do this without actually moving into the directory. Normally we don't specify a path to `rm` and it assumes that we mean in the current directory. However, we can also tell it to remove files from a directory that we're not currently in. Remove everything from the directory you made with `rm <directoryname>/*`. Similarly, we can do a "remote ls" to see if that command worked: `ls <directoryname>`. Once your directory is empty, try removing it again with `rmdir`.

Your Turn

Congratulations on making it through the tutorial! Time to put your directory creating, file creating, and change directory skills to work. Go ahead and create this hierarchy of files for practices, and remember the shortcuts:

- ~ for home directory
- . for the directory you're currently in
- .. for the directory above the directory you're currently in

```
Pets
  |---> Fish
  |        |---> Saltwater
  |        |---> Freshwater
  |
  |---> Mammals
  |        |---> Dogs
  |        |       |---> Large
  |        |       |---> Medium
  |        |       |---> Small
  |        |
  |        |---> Cats
  |                |---> Longhair
  |                |---> Shorthair
  |
  |---> Reptiles
           |---> Snakes
           |---> Turtles
```

Retrieved from "http://wiki.sc-education.org/index.php/Fun_with_Linux"

- This page was last modified on 4 April 2008, at 18:22.