22S:166
Lab session 1
Using Linux and the text editor

Aug. 28, 2009

1 Getting started

Log onto the lab computers using the username and password provided to you by the Computer Support Group.

Left click the icon of a computer screen at the bottom of the screen to bring up a terminal window. If you are using the Linux network for the first time, change your password by entering

```
passwd
```

and then following the prompts.

2 The course home page

Bring up the Firefox Internet browser by clicking on the icon of the world with a flaming fox on it.

Type in the URL (address) of the course homepage:

```
www.stat.uiowa.edu/~kcowles/st66_2009
```

Go to the "Web resources" page, and notice links to reading assignments. You should complete the readings by the end of the week under which they are listed. For example, you should finish reading "Learning the shell" from the Linux Command link by today.

3 Using the Linux shell

The Linux shell passes the commands that you type to the operating system. There are actually a choice of several different shells under Linux. The system administrators at the Computer Support group have made tcsh the default shell on our network, and that is what we will use in this class.

The Linux shell has hundreds of commands. However, most basic tasks can be accomplished using only about a dozen of them.

Linux commands and filenames are case-sensitive; e.g. "LS" is different from "ls."

3.1 ls

- probably most often-used command in Linux

- lists contents of a directory of the current directory unless you tell it otherwise
- hidden files
  - files whose names start with a period are "hidden"; e.g. .cshrc – the configuration file
  - use ls -a to include the hidden files in the listing
- use ls -l to get the "long listing," including security mode

Enter the command to list the contents of your home directory. Use the up-arrow to recall the line that you just typed. Add the option on the end to include the hidden files. Use the up-arrow again; backspace to remove the previous option and add the one to get the detailed listing.

3.2 more

- used to view the contents of a file
- syntax: more <filename>
- got its name because after displaying each page of text in the file, it pauses and prints "--More--" at bottom of screen
  you hit spacebar to see next screenful
- stops and returns you to Linux prompt when end of file is reached
- typing q also stop more and returns you to the Linux prompt

Enter the command to display the contents of your configuration file.

If you try to use more on a file that is not a text file, you will get strange results, as we will see later.

4 Text editor

Now we will use a text editor to create two text files in the home directory. If you already know a Linux text editor such as Vi, Vim, or Emacs, you are welcome to use it for this course. If not, please learn to use the Gnu text editor called gedit. To invoke it, enter gedit

We will type a file with the names of dogs. If you have one or more dogs, type in their names, one on each line. If not, type in the names of some of my dogs:

```
Kirri
Lucy
```

We now will use the File menu and the "Save as" option to save this file in the home directory under the name "dogs." If you already have a directory called "dogs" inside your home directory, you will have to choose a different name.

Use gedit again to create a text file named "cats." If you have one or more cats, enter their names. Otherwise put in a single line with the word None

Save the file. Use the appropriate command to list the files in your home directory, so you can make sure that the two new files are there. Then use the appropriate command to view the contents of the dogs file.

5 Linux shell commands, continued

5.1 mkdir

- short for "make directory"
- creates a new directory
  in the current directory unless you tell it otherwise

Create a directory called "pets" in your root directory.

5.2 mv

- moves or renames a file or directory
- syntax: mv <source> <destination>
- examples:
  - mv dogs pets
    moves the dogs file out of the home directory and into the pets directory

    Use ls to verify that dogs is no longer in the home directory.
  - mv cats felines
    renames cats to be called felines

    use "ls" to make sure it worked
- Now rename the file back to "cats."
- Now issue the command to move the cats file into the pets directory

5.3 cd

- changes the active directory
- syntax:
  cd <directory name>
- example: cd pets
  - if the directory you want to change to is in the current directory

- cd /group/ftp/pub/kcowles/datasets
  - need "pathname" to go to a directory that is elsewhere
  - This is the directory that the "Datasets" link on the course web page accesses
  - List its contents using the long form. Interpret the security mode for one of the files. Is this what you would expect for a file in this directory?
  - Enter more campaign_sample.zip to see what a non-text file looks like when you try to display it with more. You may need to close the terminal window and open a different one after doing this.
- cd by itself with no directory name will return you to your home directory; do that now.

5.4 pwd

- prints the current (working) directory
- syntax:
  pwd

Your home directory isn't really the top of the tree – there are other higher directories on the server where our home directories reside.

You can tell this because there are lots of parts to the pathname of directory. The slashes separate the names of the levels of the tree.

To go back to the directory immediately above the current directory, use cd .. (the two periods are part of the command).

Pathnames enable you to carry out operations on directories other than the current one. A *relative pathname* assumes that the path of the current directory goes before the part that you type. For example, from in your home directory, enter

```
ls pets
more pets/dogs
ls /group/ftp
```

The last row above contains an *absolute pathname*. Note that it starts with a slash.

Suppose that we decide to organize our "pets" subdirectory differently. We want to have individual subdirectories called "mammals," "reptiles," "birds," and "Other" in the pets directory, and we want the files called "dogs" and "cats" to be in the "mammals" subdirectory.

To practice using pathnames, let's create the new subdirectories and move the files while keeping the home directory as the current directory.

```
mkdir pets/mammals
mkdir pets/reptiles
mkdir pets/birds
mkdir pets/other
```

5.5 cp

- cp copies a file or directory

Just to be on the safe side, let's copy "dogs" and "cats" from the pets directory into the pets/mammals directory. After checking that the copies are there and correct, we will delete the originals.

```
cp pets/dogs pets/mammals
cp pets/cats pets/mammals
```

Then issue the commands to list the contents of the subdirectory pets/mammals and to list the contents of the files "dogs" and "cats" in that subdirectory.

5.6 rm

- rm removes a file
- note: Linux does not have an un-remove command, so use rm with care.

Finally, use rm to remove the originals of dogs and cats.

```
rm pets/dogs
rm pets/cats
```

The "mv" and "cp" commands can be used on directories as well as files. The -r option is needed to make cp copy a directory. To make a copy of the pets directory that is called "animals," from your home directory enter

```
cp -r pets animals
```

Use the appropriate commands to verify that the new directory is there, and that it contains the same subdirectories and files as "pets."

5.7 rmdir

- rmdir removes a directory

We don't need two copies of the pets directory. Note that only empty directories can be removed.

```
cd pets/mammals
rm cats
rm dogs
cd ..
rmdir mammals
rmdir reptiles
rmdir birds
rmdir other
cd ..
rmdir pets
```


Just for practice, let's rename the "animals" directory back to "pets"

```
mv animals pets
```