CMAS Training – UNIX Basics

Basic commands

A command is simply a program that tells the computer to do something. In Windows, you usually start programs using an icon; in UNIX, you start programs by typing a command. Most of the commands you will need for this class have the form:

command *argument*

The *argument* indicates what the command should perform its action on, usually a file. Some commands don't require an *argument,* while other commands may need more than one. Commands in UNIX are case sensitive; **command** and **Command** are not the same.

Here are some commands you'll probably use in this class:

Command	***Action***
man *command*	Get help for the given *command*
ls *directory*	List the contents of *directory*; if no *directory* is given, **ls** lists the contents of the directory you are currently in
cd *directory*	Change to *directory*
pwd	Print the path of the directory you are currently in
cp *file1 file2*	Copy *file1* to *file2*; **cp** will overwrite *file2* if it exists
mv *file1 file2*	Move (rename) *file1* to *file2*; **mv** will overwrite *file2* if it exists
rm *file*	Remove (delete) *file1*
head *file*	Display the first 10 lines of *file*
tail *file*	Display the last 10 lines of *file*
more *file*	Display *file* one page at a time
grep *pattern file*	Search for the given *pattern* in *file*
setenv *name value*	Assign *value* to the environment variable named *name*
echo $*name*	Print the value associated with the environment variable named *name*; note that the $ character immediately precedes *name* (no space)
command1 | command2	Send the output from **command1** to **command2**; usually **command2** is the **more** command to display the output from **command1** one page at a time
ncdump *file*	Dump the NetCDF file *file* as text
pave -f *file1* -f *file2* … -f *file*n	Run **pave** and load in *file1*, *file2*, … *file*n; you must give the full path and file name for each file loaded

Files and directories

Files in UNIX are organized into directories, similar to folders in Windows. While Windows has the base directory **C:\**, the UNIX base directory is just **/**. The files you will use in this class are located in the directory **/home/training/smoke** for the SMOKE course and **/home/training/cmaq** for the CMAQ course. You will use the **cd** command to move into different directories and the **ls** command to list the files in a particular directory.

One useful construct in UNIX is the concept of relative paths. The parent of any directory can be referenced as "**..**". For example, if you are currently in the directory **/home/training/smoke/data/ge_dat** and want to change to the directory **/home/training/smoke/data/inventory**, you can use the command:

```
cd ../inventory
```

This command indicates that you should first go up one directory (into **/home/training/smoke/data**) and then go into the **inventory** directory.

When providing file names as arguments to UNIX commands (i.e. **more** *file*), you must tell the system exactly where *file* is located. One way to do this is to use the full path (also called the absolute path) and file name. Suppose you want to look at the **scc_desc.txt** file in the **/home/training/smoke/data/ge_dat** directory. Regardless of what directory you are currently in, you could use the following command to view the file:

```
more /home/training/smoke/data/ge_dat/scc_desc.txt
```

Most UNIX commands will look in the current directory when just given a file name. So, if you are already in the **/home/training/smoke/data/ge_dat** directory, you can just use the command:

```
more scc_desc.txt
```

Another option is to use relative path names. If you are in the directory **/home/training/smoke/data**, you can use the command:

```
more ge_dat/scc_desc.txt
```

You can also use environment variables to store directory or file names. For example, the environment variable SMKROOT is set to **/home/training/smoke**. This means you can use the following command to view the **scc_desc.txt** file:

```
more $SMKROOT/data/ge_dat/scc_desc.txt
```

This last command is the same as using the absolute path and file name, so it can be used from any directory.

Useful shortcuts

The up and down arrows on the keyboard allow you to scroll through previously used commands. If you listed the contents of a directory with **ls** three commands ago, you can hit the up arrow three times to get back to that command. While scrolling through the commands, the up arrow shows you older commands and the down arrow shows you more recently used commands.

The Tab key can be used to complete file and directory names while you're typing commands. Suppose you have a directory containing three files:

bfac.summer.txt   bfac.winter.txt   costcy.txt

If you want to view the **bfac.summer.txt** file with the **more** command, you can type the following command:

```
more b<Tab>
```

where <Tab> means press the Tab key. The computer will automatically fill in the command as far as it can:

```
more bfac.
```

Since there are two file names that start with **bfac.**, the computer doesn't know which one you want, so you'll need to type more of the name:

```
more bfac.s<Tab>
```

Now the computer can fill in the rest of the file name to complete the command:

```
more bfac.summer.txt
```

If you wanted to view the **costcy.txt** file, you could simply type:

```
more c<Tab>
```

The file name would get filled in completely since there is only one file in the directory whose name starts with the letter "c". Using the <Tab> key to complete file names is especially useful for long file names since you can avoid typing the whole name (and making typos).

The computer you are using for this training also allows you to copy and paste text using the mouse. If you highlight text with the left mouse button, you can paste that text by pressing the middle mouse button.