

Why do you need to learn UNIX?

1. B/c the world's most powerful computers use it.
2. B/c a lot of software you need is freely available for UNIX systems
3. B/c SGIs don't run Windows and you will need to use SGIs during this class (and throughout your career).

What are the advantages of learning how to use it?

1. It's portable: UNIX is used on many, many computers and it's nearly the same, even on different operating systems.
2. It features powerful applications and software.
3. You can save yourself a lot of time and careless mistakes in the long run by writing scripts and programs to help you analyze your data.

What are some key differences between Windows and UNIX?

Windows has a nice graphical interface. To run a program in Windows, you double click an icon. UNIX uses primarily a text-based display. There are few icons in UNIX. UNIX does have a graphical interface, but its real power lies in the command line system. To run a program, you must enter a command. It can be intimidating to learn the necessary commands required to run the programs you want in UNIX. This primer will help you get started.

- I. Logging in
- II. Text editors
- III. Commands
- IV. Configuring your shell: .cshrc
- V. Redirection & pipes

I. Logging in

- A. System administrator will give you a login name and password. Both are case sensitive – anita, Anita, and ANITA are all different
- B. Change your password: At the prompt (the % sign is the default prompt), type
% passwd
You'll be asked for your old password and then asked to create a new one. Don't forget your new password!

II. Text editors

- A. To create files, you need to be able to write and save text or data in a file. The easiest way to write a text file in UNIX is using nedit. At the prompt type
% nedit
An X window will open with the program nedit. You can write a text file just like you would using WordPad, Notepad, or even Microsoft Word. Type the following:
This is a test of my text editing skills.

When you have finished typing, click File – Save As – and give the file a name, **test**. It's a good idea to add an extension (.doc, .txt, .mac, .jpg, etc.)

to the filename you're working with so that you know what application it will be used for, so **test.txt**.

- B. Another handy and powerful text editor is **vi**, but it is more difficult to use than **nedit**.
- C. Some guidelines in naming files:
 - i. Capitalization is important – data, DATA, and Data are 3 different files.
 - ii. You can not have spaces between words, i.e. **test file**. You must write the filename as one word or two words separated by something like **_** or another character, such as **test_file**.
 - iii. Do not use the following characters in a filename: () , . - ! ? * \$ # @ ^ &

III. Commands

- A. There are many, many commands one can use in UNIX. There are so many that UNIX may initially seem very frightening. A Windows-like environment is often easier to use because we can at least see what our options are. In UNIX, we can get by very nicely knowing only a few commands, but these commands must be put to memory. Here are a few of the most useful:

ls	lists the contents of a director or subdirectory
more	shows the contents of a file, page bye page
cd	change directory
mkdir	make a new directory
rmdir	remove a directory
rm	remove a file or directory
pwd	print working directory
cwd	current working directory
mv	move a file somewhere
cp	copy a file somewhere
who	tells you who you're logged in as
date	displays the date and time
cal	displays the month's calendar
man	gives you the manual
tar	
gzip	zips a file
gunzip	unzips a file
ps	prints all the processes running on your computer
kill	kills a specific process
grep	grabs a specific line of text from a file or directory
*	use this symbol as a wildcard

At the prompt **%** type

% ls

and the contents of your directory will be displayed. You have no files in it except for the one you created with nedit.

To view the contents of the file, type

% more test.txt

and you will see what you have written for the file test one page at a time.

To scroll down press the space bar.

Other ways to view the contents of files:

% cat test.txt

% head test.txt (by default gives first/last 10 lines)

% tail test.txt

% head 20 filename (gives first 20 lines)

How to print the file?

% lp test.txt

% lpstat (gives status of print requests and their request numbers)

% cancel laser1 -124 (cancels print job – cancel printername printer request number)

What directory are you in? At the prompt **%** type,

% pwd

and `11/users/akishore/` should appear. You are in this directory.

Create a new directory for this class, BCMB8190. At the prompt **%** type

% mkdir bcmb8190

Then type

% ls

The following should appear:

test.txt

bcmb8190

Now change your present directory to bcmb8190. Type

% cd bcmb8190

% pwd

And `11/users/akishore/bcmb8190` should appear. You are now in this directory. To see the contents of this directory, type:

% ls

But nothing should appear because you don't have any files in this directory. If you make a mistake and want to delete the directory, type

% rm bcmb8190

If the directory is empty, contains no files, rm will remove the directory.

If you want to remove all the contents of the files as well as the directory, type

% rm -r bcmb8190

BEWARE: **rm** will remove your files permanently. There is no way to get them back. In order to prevent accidental deletion of important files, get in the habit of typing

% rm -i filename

The **-i** flag asks you if you really want to remove the files, yes or no.

Make a subdirectory called data in your directory to store NMR data.

% mkdir data

You may want to change the name of your file. You can do this by either copying the file to another name or changing the name directly.

% cp test.txt test2.txt

% mv test.txt newtest.txt

mv completely writes over the file. If you have something with that same name, it will be changed forever.

Shortcuts to move in and around directories:

% **cd ~/** means change directories to your (user's) home directory
% **cd ./** means look with the present directory
% **cd ../** go up one level in the directory structure

Copy the file procpair located in 11/users/akishore/data/1H_test.fid/ to your directory:

% **cp 11/users/akishore/data/1H_test.fid/procpair .**

provided you are already in the right directory /usr/yourname/data. If you're not already in your data directory type the following:

% **cp 11/users/akishore/data/1H_test.fid/procpair ~/data**

B. Introduction to the Command Line

i. To search through files to find specific instances of text

% **grep reffrq procpair**

To run this command you need to be in the directory where the file **procpair** is located. Go to this directory first, then run the grep command.

This command searches the file procpair one line at a time for the instance of the word reffrq and prints the line containing the word reffrq to the screen.

% **grep -v reffrq procpair**

Searches the file procpair one line at a time for lines that DO NOT contain the word reffrq and prints output to the screen.

C. Sorting files

% **sort filename1 > filename2**

A handy way of sifting through data, especially if the data is in lists. Similar to what you might do in Excel. The above command sorts through the old file, arranges the items in alphabetical order and writes the newly ordered items to filename2. The > sign is like a funnel that sucks up the output and sends into to the input file through a chute.

% **sort -rn filename1**

The -rn flags do two things: -r sorts filename in numerical order; -n sorts in numerical order. Combined the command sorts in reverse numerical order.

% **sort +1 filename1**

This skips column one and continues to sort

D. Finding files

% **find /11/users/akishore/ -name unix -print**

This command looks in the specified directory for the filename and prints the location to the screen. Similar to the Find or Search function in Windows. If you have many files to look through to find the specific one, run the job in the background. (see below)

E. Running jobs in the background – use the ampersand (&)

% **command -option argument &**

This sign (&) is especially important if you're running long jobs but you still want to have access to the shell. It's also good when running netscape:

% netscape&

F. Finding continued...

What if there are many files that we want to find? We can send the results of the search to another file, not to the screen using the redirection tool. Running this command in the background also allows us to continue working in the same screen.

% find 11/users/akishore -name filename -print > results &

F. Data transfer

At some point you will have to transfer your data from one computer to another, for example the spectrometer computer to a local workstation. For a simple, small text file, simply use ftp. If the file you want to transfer is large or you want to transfer many files at once, tar the files first, then zip them, then transfer them. To transfer NMR data it is a good idea to tar and zip the files. For example

% ls

data/ bcmb8190/ test.txt proton.fid/ phosphorus.fid/

This directory contains several folders and one text file. I want to transfer proton.fid and phosphorus.fid from the spectrometer to my computer. These directories are not simple files. I could individually transfer the contents of the directory to my computer, but there is a faster way. First tar the files:

% tar cvf ak.tar proton.fid phosphorus.fid

where ak.tar is the name of temporary tar file created that stores the contents of the *.fid directories

% ls

You will see all the directories from above plus ak.tar. Now make this tar file very small and easy to transfer by zipping it:

% gzip ak.tar

% ls

You will see all the directories from above plus ak.tar.gz. Now you are ready to transfer the data to your computer using ftp.

% ftp

ftp> open IP address (128.192.9.235)

login

password

% bin

% put ak.tar.gz

bin ensures you are in binary mode to transfer files so your data does not get corrupted.

put transfers a file from computer A to B, where A is the local computer and B is the remote computer.

get transfers a file from B to A where B is the remote computer and A is the local computer.

G. Options and arguments

Commands alone are not enough. You can specify options to run a command on your file or program. This is where UNIX becomes very confusing because the number of options seems to be infinite. Here are some important options that you might use frequently:

ls -a list all files, including normally hidden files

ls -l list long format info on files

ls -F list directories and files slightly differently so you can distinguish

ls -x list files in ASCII order: numbers, uppercase letters, lowercase letters
ls -R list files in reverse ABC order
ls -n list files in numerical order
ls -t list most recent files first
ps -u display all processes you are running
ps -ef display all processes all users are running
kill -9 filename kill it completely

H. Changing permissions – you may not need this right away!

Most files that you create, you have full access to read, write to, or execute. However, you may want to block other people from reading, writing, or executing your files. Or you may have files that you cannot read, write to, or execute b/c someone has blocked you from them. To deal with this problem you need to change permissions on the document.

To change permission, first see what the permissions are on a document:

% ls -l

The first column is the most important. It consists of 10 characters. The first one indicates the type of file (ordinary – or directory d). The remaining 9 characters define the permissions of the user (first 3), group (middle 3), all users (last 3). R = read, w=write, x = executable. A dash means permission not granted. To change permissions type

% chmod u+x filename

This changes the mode of the user and adds permission to execute a file. U = user g = group, a = all. + = adds - = subtracts.

IV. The shell

You don't communicate directly with the computer. You tell the shell what to do, and it in turn tells the kernel what to do. You give the shell instructions in the form of environment variables and commands. You will probably need to define or adjust the some variables:

A. Aliasing, or shortcuts

Shortcuts can be created in your .cshrc file so that you don't have to type a lot to perform an often repeated task. For example, you can quickly connect to a computer using a shortcut if you add it to your .cshrc file. Your .cshrc file will be found in your home directory, or the directory where you are when you log in. To add a line to your .cshrc, type

% nedit .cshrc

You can add the following line to your .cshrc file

alias chem. 'ssh yourlogin@sunchem.chem.uga.edu'

NOTE: Every time you make a change to the .cshrc file, you must source it – sort of like starting over without rebooting. At the prompt type

% source .cshrc

If there are problems with the file, errors will immediately appear. Use the errors to troubleshoot. If you know one line in the program works, put a comment sign (#) before a line that you know doesn't work and test each line one by one until you figure out which line is causing the trouble.

An alias for the really lazy.

% **alias x 'exit'**

This creates the shortcut x that when entered by itself, exits the shell.

B. Status of jobs

How can you check if your job is running?

% **ps -u**

This command will display all processes that you are running. But before you submit a big job, you should check if others are also running jobs on the same computer.

Remember UNIX is a multi-user system, so just b/c someone isn't sitting at the same computer doesn't mean someone else isn't running a job remotely. To check other processes running on the same computer

% **ps -ef**

This will display all jobs running on the computer where you are logged in with the process ID number (PID).

How can I terminate my job?

You may want to terminate a job or processes, i.e. cancel printing, stop NMRPipe, kill an Amber job, if you see that it is not running correctly or you want to change something.

Note the PID from the ps command and type at the prompt

% **kill -9 PID** - but type the actual process ID number in the command line, not PID

Another way to force quit programs:

% [**Ctrl-C**] (type ctrl and C at the same time)

V. Redirection and Pipes

- A. Computer programmers love to save time and keystrokes, so they have created a number of shortcuts. One shortcut is to send the output of one command directly into the input of another one. This input/output can be accomplished by a) redirection, using > or < signs or b) pipes. (We have already used both the redirection and pipe tools in previous examples.) In redirection, the greater than sign, > acts like a funnel that swallows the command on the large side of the sign and sends the output to the small side of the sign.

input data command output data

% **ls *a > fileswitha**

This lists all files that have *a (* is wildcard) and sends output to a file called fileswitha.

The redirection can go either way

Command > filename	sends output of command to filename
Command < filename	let filename input be used by command
Command >> filename	output of command is appended to end of filename
Command1 command2	run command1 then send output to command2

% **ls *a | grep data | lp**

Way of combining several commands in one step. Above lists all files with *a, takes those files and looks for the instance of data in all of them and sends the output (the lines of those files) to the printer.

Errors: pay attention to the errors the computer gives you! Can direct errors to the screen or to a file. The latter is especially good for long jobs with many errors, something like a log file.

```
% ls *a &> errorak
```

B. Examples

i. You will frequently need to extract a limited number of parameters from a long file. Rather than search through the file by hand, it's much easier and more reliable to ask the computer to search through strings for a particular incidence of text. For example, to process your NMR data, to successfully transform the fid into a Fourier transformed spectrum, you will need certain parameters from the propar file located in your .fid directory. The following command will allow you to extract the selected parameter to a text file for viewing later.

ii. **% grep sfrq > params.txt**

iii. This command grabs (using grep) or looks for the instance of the word sfrq and takes the whole line in which each of these words appears and sends the lines to a file called params.txt. You can then view the file params.txt to see the important information you wish to extract from the propar file. Another way to do this is to open the file **propar** using the more command.

```
% more propar
```

Use the space command to scroll through the text. To search for a specific word or number type

```
% /reffrq [enter]
```

and you will be sent to the first line with the text reffrq

C. Pipes are used extensively in NMRPipe, an NMR data processing package that uses the piping system of UNIX to send input and output back and forth quickly and easily. You will learn more about NMRPipe scripts in an upcoming lab session.

References:

- Teach Yourself UNIX 3rd edition, 1995, Reichard & Johnson
- <http://www.ugu.com/sui/ugu/show?I=help.articles.unix101&F=1111111111&G=Y>
- <http://www.eits.uga.edu/wsg/unix/unix.html>