Statistics 243: *class notes*

William J. De Meo

August 29, 1997

Topics

1 Shell Services

We begin with some useful examples. The command
`head -12 filename`
will show you the first 12 lines of the file called *filename*. The command
`tail -14 filename`
will show you the last 14 lines of the file called *filename*. The command `grep` finds regular expressions in its input. For example,
`grep string filename`
displays those lines of *filename* containing occurances of *string*.

1.1 Redirection

To use `grep` to find any errors in the output of *program*, we redirect the program's output, making it input to the `grep` command. To do this, run *program* as follows:
`program |grep -i error`
To learn about the kind of expressions `grep` will accept, check out the article on regular expressions on Phil Spector's web page.
Some more examples of redirection:
`> &` redirects both stderr and stdout to a file
`> > &` appends both to file
`|&` pipes both to a file
To seperate stdout and stderr you could write
`(command > outfile) > & errorfile`
Once you direct stdout to `outfile`, all that's left is stderr which you can redirect into `errorfile`.

Suppose you want to execute a command on a bunch of files whose names are all listed in the file filenames. For example, suppose you want to copy all files in filenames to `someotherdirectory`. Use the ‘ charater:
`cp ‘cat filenames‘ someotherdirectory`
Another example:
`emacs ‘ls -t |head -5‘`
This opens emacs with 5 buffers containing the last 5 files.

1.2 Job Control

`ps` lists processes you've initiated in a given shell.
If you run command on some shell on machine bugaboo, then run `ps` in a different shell window on bugaboo, it won't show you the job command running on bugaboo.
`ps -af` will list all processes on bugaboo.
`ps -af | grep` *`username`* will list all processes on bugaboo which have been initiated by username.
If you run a job that will run for a while and that you will check periodically, you must remember on what machine you started the job.

In an interactive program
```
C-c
```
interrupts[1]. Some programs don't listen to this so the next thing you could do is
```
C-\
```
which submits a quit signal. Many programs won't quit at optimal times since this is an order to quit immediately. The suspend signal
```
C-z
```
puts job in the backround. To run any job in the backround, put an & at the end of the command. This does not put output in backround, so any output will still be printed to terminal screen. When you put a job in the backround you should, almost invariably, redirect stdout. If you will remain at the terminal use
```
command > output &
```
If you will leave the terminal, you have to also redirect stderror with
```
command > & output &
```
You could also redirect output to other shells. Type tty to find out what filename corresponds to a given shell, then redirect output to that filename. Remember not to kill the shell!

If a job is running for a while and you want to leave, use `C-z` and you get
```
[2] command (stopped)
```
Now type `bg` to get the job to start again in background. To get it back into the foreground type `fg`. If there are other jobs running, be more specific by typing `fg 2`, where 2 is the number of the job. Job numbers can be found by typing `jobs`.

If you logout and then want to kill your job, get back on machine on which job started. Type `ps -af | grep` *`username`*. Note the process id (pid), say 27531, then type
```
kill 27531
```
Easiest way to check that it worked is to type it again
```
!!
```
If it's still running, you need something stronger. Try
```
kill -9 27531
```
If it still doesn't stop, it's a zombie, and the only way to stop it is to shut down the machine.

Miscellaneous note:
If you want a filename but don't really want a file you could use /dev/null and nothing happens.

1.3 Path Names

When Phil puts samples in the s243 directory, it will be in the ta's directory. That is,
```
/class/g/s243/s243/samples
```
If you want to copy file `test.c` from this directory you would write
```
cp /clas/g/s243/s243/samples
```
But it's a pain to do this every time, so we need some shortcuts.
. is the current directory
.. is one level up
So from your home directory for the class account, you can type
```
cp ../s243/samples/test.c
```

[1]Notation: C-c means that you must hold down the Control key and press the letter c

Another Example: What if you created a program in your home directory, but then do some clean up work and want to move it into it's own directory. Enter the following commands:

```
mkdir hw1
cd hw1
mv ../program.c
```

Your home directory is denoted $\sim$, which is a shell service.