Running Jobs in the Background on UNIX Machines

It is important to run jobs in the background to free up the terminals in department's 4th floor lab. Here we briefly describe some simple procedures that will do this.

In order to run a job in the background, type the following command:

progname args **<** *inputfile* **>** **&** *outputfile* **&**

where:
progname is the name of your program, *args* is one or more optional arguments you need to give your program, *inputfile* is the input file that will be executed and *outputfile* is a file that contain the output.

But, when you logout your job will be terminated even if it is running in the background.

If you want to run a job so that it continues even AFTER you LOGOUT, type the following:

nice nohup progname args **<** *inputfile* **>** **&** *outputfile* **&**

where, *nice* means the program runs at lower than interactive priority.
This will allow you to have your *outputfile* generated after successfully executing *inputfile*.

Example:

For MATLAB, an example is as follows:

matlab **<** *my_prog.m* **>** **&** *output* **&**

The output that will be otherwise printed on the screen in interactive mode now will appear in the file *output* after the execution.

Similarly for the run with *nohup* , use the following command.

nice nohup matlab **<** *my_prog.m* **>** **&** *output* **&**

It is IMPORTANT to ELIMI NATE any graphical output throughout the execution. So, if your inputfile, i.e *my_prog.m* , includes graphical output, type

unsetenv DISPLAY
nice nohup progname args **<** *inputfile* **>** **&** *outputfile* **&**

Syntax for TSP and STATA

To do the same thing with TSP use the following command

```
tsp < my_prog.tsp > & output.out &
```

or,

```
nice nohup tsp < my_prog.tsp > & output.out &
```

To do the same thing with STATA, use the following command

```
stata < my_prog.do > & output &
```

or,

```
nice nohup stata < my_prog.do > & output &
```

Similarly for SAS

```
sas < my_prog.sas > & output &
```

or,

```
nice nohup sas < my_prog.sas > & output &
```

Killing Jobs in the Background

If you want to terminate the program, first thing to do is to identify the process. When you run your program in the background, the process id will appear after the command line.

For instance, running *my_prog.m* in the background will display the following:

```
rochelle.eco.utexas.edu{username}94: nice nohup matlab <
my_prog.m > & output &
[1] 28465
```

where *28465* is the process id.

However, you can identify your processes at any time by typing:

```
ps
```

Following our example, it leads to the following display.

```
rochelle.eco.utexas.edu{username}99: ps
   PID TTY      TIME CMD
 28514 pts/19   0:00 ps
 28090 pts/19   0:00 tcsh
 28465 pts/19   0:02 matlab
```

Then killing this process is achieved by:

```
kill 28465
```