

**ECU178 Computer Science:  
207SE - Operating Systems, Security and Networks  
Coursework**

Due on March 16th 2015

**Robert Rigler : 4939377**

## Contents

<b>Week 12: Multitasking vs Multiprogramming</b>	<b>3</b>
Multiprogramming . . . . .	3
Multitasking . . . . .	3
<b>Week 14: Process Manipulation &amp; Nohup</b>	<b>4</b>
Process Manipulation . . . . .	4

## Week 12: Multitasking vs Multiprogramming

In this task I am going to be comparing two different types of process scheduling: Multitasking, and Multiprogramming. I will look into what they are, their differences and their similarities.

### Multiprogramming

**Definition:** A way of scheduling processes to maximise CPU usage by switching processes that are 'waiting' for I/O, it ensures that the CPU is never idle.

Much older systems, unlike modern computers were very expensive and slow and often, when a process needed to use a peripheral device It often meant that the CPU was sitting idle for a long period of time. The solution to this is 'batch processing'.

Multiprogramming allows a computer to do several tasks at the same time. When a group of processes are marked 'Ready' for execution they are placed in a queue in main memory. The first process from this queue is then loaded into the CPU and is executed. There may come a time when this process is interrupted because It needs I/O to continue. At this point the process changed to a 'waiting' state. The process is then swapped out of the CPU into the I/O queue, and the next process in the 'Ready Queue' is swapped into the CPU. When the I/O request of the first process is completed, it is then placed back into the 'Ready queue'. This cycle continues until there are no jobs to be processed.

### Multitasking

**Definition:** A logical extension of Multiprogramming, it involves rapidly switching between processed in the 'Ready state' to give the impression that they are all running simultaneously.

In Multiprogramming, processes are executing one at a time, in the order that they are placed into the ready queue. This means that only one process can be actively used at a time. Similarly in multitasking, processes are executed individually, but ther is also a certain level of concurrency; Because once a process has used it allotted processing time, It is swapped back into main memory.

This is beneficial, because with multiprogramming, a process has complete control over the CPU until an interrupt is called. There may be a situation where a process does not call an interrupt and takes a long time to finish processing. This will cause shorter, more time efficient or more important processes to be delayed until the first process is finished.

## Week 14: Process Manipulation & Nohup

### Process Manipulation

For this task I will look into the different ways to manipulate a process, and show examples of how to use each command.

Command	Description
<i>command</i>	Type the name of the process to start it
<i>command &amp;</i>	Start the process in the background (symbolised by the & symbol)
<i>ps -au</i>	Shows all the processes currently running on the machine
<i>ps -ux</i>	Shows all the processes currently running owned by the current user
<i>jobs</i>	Shows the processs that are currently suspended.
<i>CTRL - C</i>	Kills the process running in the foreground
<i>kill -9 x</i>	Kills the process with the PID <i>x</i>
<i>kill %1</i>	Kills the process with job number <i>1</i>
<i>CTRL - Z</i>	Susoends the process curently running in the foreground.
<i>kill -cont %1</i>	Continues the execution of suspended job %1
<i>bg %1</i>	Pushes job number 1 to to the background
<i>fg %1</i>	Pushes job number 1 to to the foreground

In the pages below, I will show two scenarios in which I use all of these commands. You will find a snippet of terminal code and an explanation of each step that was taken.

## Listing 1: Scenario 1

```
1 Script started on Thu 12 Mar 2015 14:58:19 GMT
2 rob@rob-HP-ProBook-6470b:$ xclock
3 ^Z
4 [1]+  Stopped                  xclock
5 rob@rob-HP-ProBook-6470b:$ jobs
6 [1]+  Stopped                  xclock
7 rob@rob-HP-ProBook-6470b:$ fg %1
8 xclock
9 ^C
10 rob@rob-HP-ProBook-6470b:$ exit
11 exit
12
13 Script done on Thu 12 Mar 2015 14:59:23 GMT
```

**This typescript recording shows how I:**

1. Starting the process *xclock* in the foreground,
2. Suspending *xclock* via CTRL-z,
3. Bringing *xclock* back to the foreground using *fg %1*
4. Finally Killing the process with CTRL-C

## Listing 2: Scenario 2

```

1
2 rob@rob-HP-ProBook-6470b:$ xclock &
3 [1] 21811
4
5 rob@rob-HP-ProBook-6470b:$ xclock
6 ^Z
7 [2]+  Stopped                  xclock
8
9 rob@rob-HP-ProBook-6470b:$ jobs
10 [1]-  Running                  xclock &
11 [2]+  Stopped                  xclock
12
13 rob@rob-HP-ProBook-6470b:$ kill %1
14 rob@rob-HP-ProBook-6470b:$ jobs
15 [1]-  Terminated              xclock
16 [2]+  Stopped                  xclock
17
18 rob@rob-HP-ProBook-6470b:$ kill -cont %2
19 rob@rob-HP-ProBook-6470b:$ jobs
20 [2]+  Running                  xclock &
21
22 rob@rob-HP-ProBook-6470b:$ ps au |grep rob
23 USER      PID     %CPU %MEM  VSZ   RSS  TTY      STAT START   TIME COMMAND
24 rob       16721   0.0   0.0  27336  4448 pts/5    Ss   14:26   0:00 /bin/bash
25 rob       21794   0.0   0.0  21892   960 pts/5    S+   15:47   0:00 script -a com2.txt
26 rob       21795   0.0   0.0  21896   396 pts/5    S+   15:47   0:00 script -a com2.txt
27 rob       21796   0.0   0.0  27224  4180 pts/15   Ss   15:47   0:00 bash -i
28 rob       21813   0.0   0.0  70556  4840 pts/15   S    15:47   0:00 xclock
29 rob       21872   0.0   0.0  22648  1320 pts/15   R+   15:48   0:00 ps au
30
31 rob@rob-HP-ProBook-6470b:$ kill -9 21813
32 rob@rob-HP-ProBook-6470b:$ jobs
33 [2]+  Killed                    xclock
34
35 rob@rob-HP-ProBook-6470b:$ exit
36 exit
37
38 Script done on Thu 12 Mar 2015 15:48:38 GMT

```

1. Starting the *xclock* process in the background,
2. Starting another *xclock* process in the foreground,
3. Suspend the *xclock* foreground process using CTRL-Z,
4. Use the Jobs Keyword to show the two *xclock* processes,
5. Kill the first *xclock* job using kill %1

6. Continue the second *xclock* process in the foreground using `kill -cont %2`
7. Show a list of my running processes using `ps -au — grep rob`
8. Finally Kill the remaining *xclock* process by using `kill -9 21813`

## Nohup

**Definition :** A command which allows a process to continue executing after the parent process has been stopped.

When running a process from the terminal or a shell