SINGAPORE POLYTECHNIC
SCHOOL OF ELECTRICAL & ELECTRONIC ENGINEERING
ET0023 OPERATING SYSTEMS

TUTORIAL 5 Process Management & Scheduling – Suggested Solutions

1.

Situation	Database access program
Process creation	Program is loaded from the disk by typing the name of the executable file at the CLI
Process runs	Process starts to run
Process is interrupted (to run another process)	A subroutine is called to setup the display on the screen
Process is waiting for an I/O to complete	A request is sent to the disk to load data into the working area
Process is terminated	User clicks "Exit" and ends the process

2. 4 ways a process can be made to terminate by the process scheduler are
   a) Normal exit (process has completed)
   b) Error exit (an error has occurred and process is terminated by the process itself)
   c) Fatal error (an error has occurred, process scheduler terminates the run)
   d) Killed by another process

   make:
   a) Automates the compilation process for large program compilation/creation jobs
   b) Ways make can terminate
      1. make terminates normally, successful completion
      2. make discovers an error, stops the run
      3. an error in the code e.g. divide-by-zero error occurs
      4. user kills the process using kill command.

   c) Yes, check the exit codes from the man
      0 make completed successfully
      1 one or more makefiles need to be rebuilt
      2 error exit (one of the makefiles failed)

   e) http://process.downloadatoz.com/tutorial/7796,find-out-kill-zombie-process-on-linux-step-by-step-guide.html

3. A non-pre-emptive scheduling algorithm picks a process to run and then just lets it run until it blocks (either on I/O or waiting for another process) or until it voluntarily releases the CPU.
E.g. processes running under MS-DOS or Windows 3

A pre-emptive scheduling algorithm picks a process and lets it run for a maximum of some fixed time. If it is still running at the end of time interval, it is suspended and the scheduler picks another process to run (if one is available).
E.g. processes running in Linux

Only the pre-emptive scheduling algorithm requires the use of a system clock. The scheduler can also use the clock to generate the necessary interrupts to instruct the CPU to switch the processes.
The non-pre-emptive scheduling algorithm does not need to.

4.

Scheduling Criterion	Desktop OS	Real Time OS	Batch Processing OS
CPU Utilization	Maximize use on foreground job	Maximize use on immediate process	Maximize use
Throughput	Maximize jobs	Ensure most pressing job done first	Maximize jobs
Turnaround time	Depends on the user, can be lower priority	Ensure most pressing job done first	Minimize time between jobs
Waiting time	Within user requirement times	Immediate	Not essential as jobs are in a batch process
Response time	Within user requirement times	Immediate	Not essential as jobs are in a batch process

5. Using the process calculation from Lect 5 pg 6
Single process = 60 sec, as each process has to wait for the previous to complete
Multitasking = 35 sec