SINGAPORE POLYTECHNIC
SCHOOL OF ELECTRICAL & ELECTRONIC ENGINEERING
ET0023 OPERATING SYSTEMS

TUTORIAL 3 – Programs and Processes Sample Solutions

1.

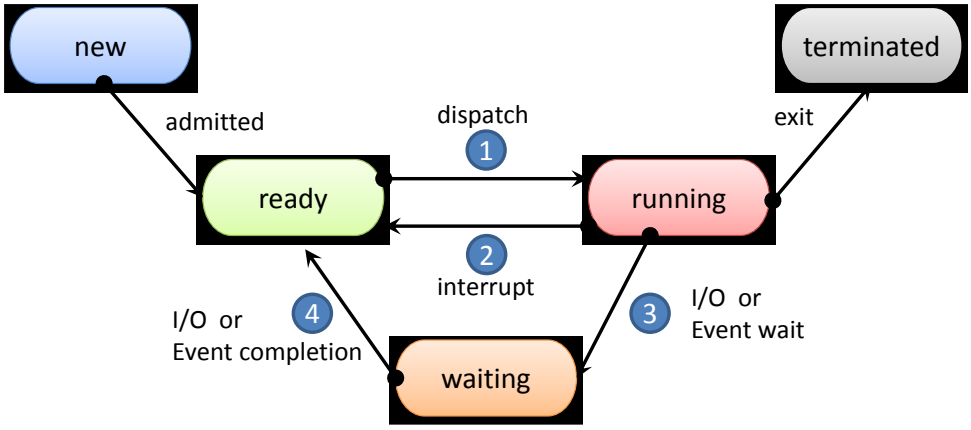
	Compilers	Interpreters
Short Notes	Usually made up of a number of programs: compiler, optimizer, linker Code is written in a text file and is fed into the compiler which converts it to intermediate code. The Linker takes the intermediate code and combines it with system libraries to form executable code which can be directly loaded and executed	Usually made up of a single program which dynamically translates what is typed in (text) to intermediate code. The program may also be saved as a text file which is then read line-by-line and translated. When the command is given, the intermediate code is executed, there is no need for a linker, and there may libraries which are required.
Example	C/C++, FORTRAN, Pascal	Basic, Lisp, Python, Perl
Final outcome	Executable machine code particular to the CPU	Intermediate code (run by a p-machine) or executable code

2. User Mode
Normal operating system mode in which all user executed programs e.g. browsers, file managers, application programs run. The program interacts with the user and not directly with the operating system

System Mode
In system mode, the program transfers control to the Operating system which enters a "privileged mode" to handle system calls e.g. File operations, calling of system functions, access to displays or input. In the system mode, the program has access to the hardware sections of the system through the operating system.

A process moves from User mode to System mode when It executes system calls e.g. to save a file, or to read a file. In these cases, the process transfers control to the system mode by creating a new process (or requesting a new process) from the Operating System. In this way, the user mode has no direct connection/interaction with the system mode.

3.



1 Dispatch – process is sent to the CPU to be executed
2 Interrupt – process is interrupted, either by external or system call within the process
3 I/O or Event wait – io operation is called, process has to wait until I/O is over
4 I/O or Event completion - io operation running, when completed it returns to process in the ready state

Missing transitions:
Waiting -> Running
A process has to transfer control to the System Mode to execute an I/O or Event. The process has to give up running (release the CPU) so that the I/O or event can take place. Hence, no such transition.

Ready -> Waiting
Will not happen, as process has to wait its turn/priority to be dispatched, it cannot move into the I/O or Event execution state

4. Take a look at this animation: http://computer.howstuffworks.com/operating-system5.htm

Co-operative Multitasking	Pre-emptive Multitasking
http://en.wikipedia.org/wiki/Computer_multitasking#Cooperative_multitasking.2Ftime-sharing	http://en.wikipedia.org/wiki/Preemption_%28computing%29
Multiple jobs are run on a single processor. Each job initially receives an equal time-slice. If job cannot complete, it can request for extension of time. Or if the job requires more time, it can take more of the time slice	Multiple jobs are run on a single processor. Each job initially receives an equal time-slice. The time-slice is managed by the Scheduler which will switch the process out regardless of whether the process requires more time. This gives a more equal running time-slice per process
Windows 3	Windows 95, XP

5. Method of compiling a C/C++ program on Linux
   a) Write and save your C/C++ program. Make sure you use a .c or .cpp extension
   b) Compile your program using gcc (for C) or g++ (C++)
      g++ -o hello hello.c
      -o specifies output file (default a.out)
   c) If no errors, you can execute your program
      ./hello
      in some cases you will need to change the attribute to executable
      chmod +x hello

hello.cpp

```
#include <iostream>
using namespace std;

int main() {
        while (1)                       // infinite loop
                cout << "Hello world!" << endl;
        return 0;
}
```

ps	display shell level process
ps a	console attached process
ps aux	display all processes running

locale for program hello or use grep command to isolate

use top to determine processes running

to kill the process, either
   a) Determine process-id from ps or top
   b) kill <pid>

or use killall <application name> e.g. killall hello