**OPERATING SYSTEMS WORKSHEET 01**

1. What are the three main purposes of an Operating System?
   1. Manages system resources efficiently
   2. Provides an interface for the user to interact with the hardware in a convenient and efficient manner.
   3. Allows the user to simulate vertical based tasks such as machines or memory.
2. What is multiprogramming? What is the main advantage of multiprogramming?
   1. Allows multiple tasks stored in memory at the same time with the CPU multiplexed among them.
3. What is time-sharing? What is the main advantage of time-shared system?
   1. Time sharing is a multiprogramming system but the user has interaction with their process (In Linux uses the & the end). Its advantageous for the use of multiple users interacting with processes.
4. In a multiprogramming and time-sharing environment, several users share the system simultaneously. This situation can result in various security problems.
   1. Two problems with this is that 1) A process can read other processes IO files. And 2) Can over utilise the system resources by running forever and using memory.
   2. Practically it may not be possible to achieve the same amount of security as a dedicated system. There could always be exploits of the system used by malicious users that may break what the system is trying to achieve.
5. What is the main difference between operating systems for mainframe computers and personal computers?
   1. A mainframe computer has a plethora of resources which need an operating system to manage the resources effectively. A personal computer may need less management of resources as there are less to manage and are more likely to aim for convenience over efficiency.
6. We have stressed the need for an OS to make efficient use of the computing hardware. When is it appropriate for the OS to forsake this principle and to ‘waste’ resources?
   1. An operating system can be allowed to waste resources in cases where convenience is more justified. Including cases for security where different protocols need to run to ensure the system is running as intended.
7. What is the purpose of interrupts? What are the differences between a trap and an interrupt?
   1. An interrupt stops an operating system from doing its task. It allows the operating system to perform a task that requires attention. A trap is a software generated interrupt which is synchronous, meaning error is in a program, whilst an interrupt is hardware generated and asynchronous.
8. What is a privileged instruction? Which of the following instructions should be privileged?
   1. A privileged instruction is an instruction decided by the operating system by including one bit (either 1 or 0) which is given to the hardware to know whether to execute it or not in certain system architectures (user or system mode).
   2. Set value of timer – Should be privileged.
   3. Read the clock – Doesn’t need to be privileged.
   4. Clear memory – Should be privileged. (Especially all memory, its process memory – yes)
   5. Turn-off interrupts – Should be privileged.
   6. Switch from user mode to monitor mode – Should be privileged.
   7. Modifying base and limit registers – Doesn’t need to be privileged.
   8. Issue a trap instruction - Doesn’t need to be privileged.
   9. Access I/O device – Should be privileged.
   10. Modify entries in device-status table – Should be privileged.
9. What is the purpose of the command interpreter? Why is it usually separate from the kernel? By what other names is it known?
   1. The purpose is to give the user with an easy way to write code for hardware and OS to execute – allowing for the system to understand what the user wants to do. It is kept from the kernel for convenience. It is also known as a compiler or assembler. Changing your command interpreter would require compiling the kernel.
10. What is the purpose of system calls? Describe three methods to pass parameters to the OS.
    1. System calls is a user interface to all the OS services. It allows for a programmer to tell the OS services to execute and to create programs.
    2. You can pass parameters through; registers, block of memory (array) or table, or in a stack.
11. Give examples of Unix System Calls for each of the following system calls major categories:
    1. Process Control
       1. Fork, exit, wait
    2. File Manipulation
       1. Open, read, write, close
    3. Device manipulation
       1. ioctl, read, write
    4. Information maintenance
       1. getpid, alarm, sleep
    5. Communications
       1. Pipe, shmget, mmap
    6. Protection
       1. chmod, unmask, chown