CSI3131 – Operating Systems
Tutorial 1 - Solution

1. What are the three main purposes of an operating system?
   - **Hardware abstraction: To provide an environment for a computer user to execute programs on computer hardware in a convenient and efficient manner.**
   - **Resource allocation: To allocate the separate resources of the computer as needed to run programs. The allocations process should be as fair and efficient as possible.**
   - **Control program: As a control program it serves two major functions: (1) supervision of the execution of user programs to prevent errors and improper use of the computer, and (2) management of the operation and control of I/O devices.**

2. Consider the various definitions of *operating systems*. Consider whether the operating system should include applications such as Web browsers and mail programs. Argue both that it should and that it should not and support your answer.
   Point: Applications such as WEB browsers and email tools are performing an increasingly important role in modern desktop computer systems. To fulfill this role, they should be incorporated as part of the operating system. By doing so, they can provide better performance and better integration with the rest of the system. In addition, these important applications can have the same look-and-feel as the operating system software.

   Counterpoint: The fundamental role of the operating system is to manage system resources such as the CPU, memory, I/O devices, etc. In addition, its role is to run software applications such as WEB browsers and email applications. By incorporating such an application into the operating system, we burden it with additional functionality. Such a burden may result in the operating system performing a less-than-satisfactory job at managing system resources. In addition, we increase the size of the operating system thereby increasing the likelihood of system crashes and security violations.

3. How does the distinction between kernel mode and user mode function as a rudimentary form of protection (security) system?
   The distinction between kernel mode and user mode provides a rudimentary form of protection in the following manner. Certain instructions could be executed only when the CPU is in kernel mode. Similarly, hardware devices can only be accessed when the program is executing in kernel mode. Control over when interrupts could only be enabled or disabled is also possible only when the CPU is in kernel mode. Consequently, the CPU has very limited capability when executing in user mode, thereby enforcing protection of the critical resources.

4. Which of the following instructions should be privileged? (**Answers in bold**)
   a. **Set value of timer.**
   b. Read the clock.
   c. **Clear memory.**
   d. Issue a trap instruction.
   e. **Turn off interrupts.**
   f. **Modify entries in device-status table.**
   g. Switch from user to kernel mode.
   h. **Access I/O device.**

5. Timers could be used to compute the current time. Provide a short description of how this could be accomplished.
   A program can use the following approach to compute current time using timer interrupts. The program could set a timer for some time in the future and go to sleep. When it is awakened by the interrupt, it could update its local variable whose value reflects the number of interrupts it has received thus far. It could then repeat this process of continually setting timer interrupts and updating the variable when interrupts are actually raised. The variable can then be used to keep track of time, particularly if initialized to some significant value. UNIX

uses such an approach by keeping a count of the number of seconds since Jan 1, 1970, in a 32 bit counter (overflow happens on Jan 19, 2038).

6. What is the purpose of system calls?
   System calls allow user-level processes to request services of the operating system. Note that system calls are implemented using software interrupts. Realize that the OS is interrupt driven and that interrupts are received both from the hardware and software (i.e. user programs) as requests to perform some action.

7. What are the five major activities of an operating system in regard to process management?
   a. **The creation and deletion of both user and system processes.**
   b. **The suspension and resumption of processes.**
   c. **The provision of mechanisms for process synchronization.**
   d. **The provision of mechanisms for process communication.**
   e. **The provision of mechanisms for handling deadlock.**

8. What are the three major activities of an operating system in regard to memory management?
   a. **Keep track of which parts of memory are currently being used and by whom.**
   b. **Decide which processes are to be loaded into memory when memory space becomes available.**
   c. **Allocate and deallocate memory space as needed.**

9. What are the three major activities of an operating system in regard to secondary-storage management?
   a. **Free-space management**
   b. **Storage allocations**
   c. **Disk scheduling.**

10. What is the purpose of the command interpreter?
    It reads commands from the user or from a file of commands and executes either directly or by spawning another process to execute a separate program.

11. What system calls in the UNIX system, have to be executed by a command interpreter or shell in order to start a new process?
    In the UNIX system, the *fork* system call followed by an *exec* call needs to be performed to start a new process. The *fork* call clones the currently executing process, while the *exec* call overlays a new program onto the new process. This is a tricky concept – the idea here is to keep in mind that the *fork* is executed in the original process, while the *exec* is executed in the spawned process. We shall study this next week.

12. What is the purpose of system programs?
    System programs can be thought of as bundles of system calls. They provide basic functionality to users so that they do not need to write their own programs to solve common problems. Common system programs include utilities to manipulate files and directories, command interpreters to execute programs (including system utilities), utilities to monitor the user base and network (UNIX who, rwho show who is logged into the local machine and in remote machines), process management, etc.

13. What is the main advantage of the layered approach to system design? What are the disadvantages of using the layered approach?
    As in all cases of modular design, designing an operating system in a modular way has several advantages. The system is easier to debug and modify because changes affect only limited sections of the system rather than touching all sections of the operating system. Information is kept only where it is needed and is accessible only within a defined and restricted area, so any bugs affecting that data must be limited to a specific module or layer.

14. Fore each of the following five services offered by the operating system, explain how each provides convenience to the users. Explain also in which cases it would be impossible for user-level programs to provide these services.
    - Program execution**. The operating system loads the contents (or sections) of an executable file into memory and begins its execution. It will allocate resources to the program during its execution (e.g. CPU). A user-level program could not be trusted to properly allocate CPU time.**
    - I/O operations. **Disks, tapes, serial lines, and other devices must be communicated with at a very low level. The user needs only to specify the device (most often treated as a file) and the operation to perform on it, while the system converts that request into device-or controller-specific commands. User-level programs cannot be trusted to access only devices they should have access to and to access them only when they are otherwise unused.**
    - File-system manipulation**. There are many details in file creation, deletion, allocation, and naming that users should not have to perform. Blocks of disk space are used by files and must be tracked. Deleting a file requires removing the name file information and freeing the allocated blocks. Protections must also be checked to assure proper file access. User programs could neither ensure adherence to protection methods nor be trusted to allocate only free blocks and deallocate blocks on file deletion.**
    - Communications. **Message passing between systems requires messages to be turned into packets of information, sent to the network controller, transmitted across a communications medium, and reassembled by the destination system. Packet ordering and data correction must take place. Again, user programs might not coordinate access to the network device, or they might receive packets destined for other processes.**
    - Error detection. **Error detection occurs at both the hardware and software levels. At the hardware level, all data transfers must be inspected to ensure that data has not been corrupted in transit. All data on media must be checked to be sure they have not changed since they were written to the media. At the software level, media must be checked for data consistency; for instance, whether the number of allocated and unallocated blocks of storage matches the total number on the device. There, errors are frequently process-independent (for instance, the corruption of data on a disk), so there must be a global program (the operating system) that handles all types of errors. Also, by having errors processed by the operating system, processes need not contain code to catch and correct all the errors possible on a system.**

    In addition to the above reasons for not having user programs provide these services directly, imaging the work it would take to develop the simplest program, if a developer had to take into consideration all the above issues.