Samuel Christopher V. Visarra
March 15, 2025
Problem Set 2

1.) Why are interrupts necessary in a time-sharing system?

Interrupts are needed to alter the normal execution of the CPU and control of the CPU is transferred from the process currently in control to the interrupting process (Mayol, 2025). A time-sharing system does this in regular time intervals (time slice) to allow for multiple tasks to run at the same time (*Difference between multiprogramming and multitasking* 2023) and to ensure that not one task monopolizes the system's resources.

2.) Why is it necessary to provide supervisor calls in in an operating system? Give an instance wherein such call has to be made.

In order to protect the system without preventing the users from using the OS facilities, memory is divided into *system's area* and *user's area*. System/supervisor calls allow the user to access parts of the *system's area*. One example is when a user needs to read a file, since the user cannot directly access the disk because it is a protected resource, it would need to make a system call in order to read that file (*Introduction of system call* 2025).

3.) Discuss one problem that is solved by double buffering, but is not properly handled by ordinary buffering.

Ordinary buffering, where a single buffer is used, faces problems related to interrupts. Allowing interrupts while a single buffer is used prevents corruption but new data might be discarded when there is still data being processed (Mayol, 2025). However, allowing interrupts might jeopardize the execution of the processes and possibly overwrite unread portions of the buffer (*ibid.*). Double buffer addresses these issues by reading from one buffer while the other is filling up (*ibid*). This works provided that the processing rate is faster than the input rate, otherwise, it would face the same problem as ordinary buffering (*ibid.*).

4.) Why was circular buffering introduced?

Streaming data, such as keyboard input, can generate lines of unpredictable lengths and using a buffer with a max size would waste space (Mayol, 2025). Circular buffering lets us handle streaming data efficiently. For example, when typing on a keyboard, a character is placed on one end of the buffer while the user is removing them from the other end (*ibid.*).

5.) Explain the statement: Spooling reduces the amount of time spent by a process waiting for an I/O operation to complete

One problem which arose in older Operating Systems was that when dealing with numerous processes, the time spent on I/O operations was much larger compared to the time spent by the

CPU on instruction execution (*Spooling in operating system* 2023). While a process provides input using an input device or while an output is being sent to an output device, the CPU would be idle (*ibid.*). To address this, spooling can be used. Spooling makes use of a buffer, otherwise known as SPOOL, which holds off jobs until a device is ready to execute those jobs (*ibid.*). This reduces CPU idle time because while the CPU executes a job, the I/O devices inputs data and outputs data to and from the SPOOL or buffer, the CPU does not need to wait for the relatively slow I/O devices and can just get the needed data for its operations from the SPOOL or buffer.

6.) Explain briefly why interrupts are indispensable in a single-processor system that supports parallel activities.

Assuming parallel activities mean multitasking, then interrupts are needed to alter the normal execution of the CPU and control of the CPU is transferred from the process currently in control to the interrupting process (Mayol, 2025). In multitasking, interrupts are done in regular time intervals (time slice) to allow for multiple tasks to run at the same time (*Difference between multiprogramming and multitasking* 2023) and to ensure not one process monopolizes the system's resources.

7.)When an interrupt occurs, the following are executed:

(1) disable_interrupts()
(2) save_pc_and_registers()
(3) identify_interrupting_device()
(4) device_handler()

Why is there a need to execute (1) and (2) before (3) and (4)?

While an interrupt is being processed, incoming interrupts are disabled to prevent a lost interrupt and so a process cannot be preempted by others (Mayol, 2025). And we would need to save the addresses of those processes so that data won't be lost and the CPU can jump back to the address to continue execution after handling the interrupting process (*ibid.*).

8.) With interrupts, one can avoid losing data that arrives in the system for processing. Explain how this is possible.

Interrupts help prevent a loss of data in an unsafe situation. An unsafe situation involves new data arriving while there is still data being processed, which would result in a loss of the newly arrived data (Mayol, 2025). An interrupt is used to force a computer to stop what it is doing, then that data of the interrupted process would be stored somewhere in memory (buffering), and the newly arrived data can be read (*ibid.*). Once execution of the interrupted process is finished, the data of the interrupted process can be taken back from the buffer and its execution can be continued (*ibid.*).

9.) An interrupt-driven operating system is usually interrupted by events that need to take control of the CPU. For example, when a set of data arrives in the system, the CPU may be

taken away from the process currently in control of it. Describe the actions of the system right after an interrupt occurred up to the time control is given back to the interrupted process.

The interrupt transfers control to the interrupt service routine through the interrupt vector which contains the addresses of all the service routines and the interrupt architecture must save the address of the interrupted instruction (Mayol, 2025). Then incoming interrupts are disabled while another interrupt is being processed to prevent a lost interrupt (*ibid.*). After the interrupting process has finished execution, control is transferred back to the interrupted process.

10.) Which of the following instructions should be privileged?

a) Set value of timer
b) Read the clock
c) Clear memory
d) Turn off interrupts
e) Switch from user to monitor/system mode

Instructions executed by the CPU can be classified into privileged and non-privileged instructions to ensure security, stability, and efficient resource management (*Privileged and non-privileged instructions in operating system* 2024). Privileged instructions are those that can only be executed by the kernel, which would need direct access to hardware or other privileged resources (*ibid.*). According to GeeksforGeeks (2024), modifying contents of the timer, clearing memory, and turning off interrupts are considered privileged instructions.

11.) Some early computers protected the operating system by placing it in a memory partition that could not be modified by either the user job or the operating system itself. Describe at least two difficulties that you think could arise with such a scheme.

Since the operating system is placed in a memory partition that could not be modified by either user job or operating system, in other words it is read-only memory (ROM). This would mean if issues were found post-production, we cannot update the operating system to address this issue and would need to replace the hardware altogether. Not being able to update post-production has several issues, one is that new hardware components won't be supported because new hardware components may need different drivers which cannot be installed in this type of operating system. Another would be software-compatibility issues, operating systems need to be regularly updated in order to satisfy the requirements of newly released applications.

12.) Show how a desire for use of control cards leads naturally to the creation of separate user and monitor (system) modes of operation.

Control cards were not intuitive to use and debugging problems that arose from using control cards were difficult to solve. Security was also another issue when using control cards, since there was no way for the computer to know whether a control card's instructions were loaded by an authorized user, the only real way to safeguard a computer was to physically secure it. This is what the eventual creation of user and system mode aimed to fix, the user mode made it

easier for the user to make use of the computer by adding layers of abstraction while the system mode created layers of security to protect the system's resources.

13.) What is buffering? How might buffering coupled with interrupts
    a) avoid loss of data in a system; and
    b) increase CPU utilization

Buffering attempts to keep both the CPU and I/O devices busy at all times by storing unprocessed data somewhere in memory called a buffer. Buffers help avoid data loss by storing the current processing data in a buffer when new data arrives via an interrupting program, this allows the processor to jump its execution back to the previous process when it finishes processing the interrupting program, otherwise, the data for the interrupted process would be lost. It increases CPU utilization because the CPU won't need to wait for an I/O device to complete an operation because it can just get the needed data from the buffer to execute jobs.

14.) Why do operating systems usually maintain a small kernel? Why not include in the kernel all the facilities available in the operating system?

A monolithic kernel is a type of kernel where all operating system services operate in kernel space, this results in having dependencies between system components (*Monolithic architecture in os* 2024). One of the disadvantages is the unreliability of this type of system, when one of the services goes down, it affects all the other services since there are dependencies between them (*ibid.*). Smaller kernels are more reliable since most operating system services run outside the kernel space, any bug or security vulnerability won't affect the entire system (*Kernel in operating system* 2025).

15.) Give one change in the system when it is upgraded from a simple multiprogramming system to time-sharing systems.

One change is that the operating system turns from multiprogramming to multitasking. Both multiprogramming and multitasking can execute more than one job, however, in multitasking a specific time is allotted to every process (*Difference between multiprogramming and multitasking* 2024). This allows for multiple tasks to run at the same time, less execution time, and an increase in responsiveness (*ibid.*).

References:

1. GeeksforGeeks. (2023, November 3). Spooling in operating system. GeeksforGeeks. https://www.geeksforgeeks.org/spooling-in-operating-system/
2. GeeksforGeeks. (2024a, August 8). Privileged and non-privileged instructions in operating system. GeeksforGeeks. https://www.geeksforgeeks.org/privileged-and-non-privileged-instructions-in-operating-system/
3. GeeksforGeeks. (2024b, September 10). Difference between multiprogramming and multitasking. GeeksforGeeks. https://www.geeksforgeeks.org/difference-between-multiprogramming-and-multitasking/
4. GeeksforGeeks. (2024c, September 24). Monolithic architecture in os. GeeksforGeeks. https://www.geeksforgeeks.org/monolithic-architecture/
5. GeeksforGeeks. (2025a, January 24). Introduction of system call. GeeksforGeeks. https://www.geeksforgeeks.org/introduction-of-system-call/
6. GeeksforGeeks. (2025b, January 24). Kernel in operating system. GeeksforGeeks. https://www.geeksforgeeks.org/kernel-in-operating-system/
7. Mayol, P. E. (2025). Basic OS Concepts Lecture SlidesFile.pdf. Cebu City.