**Theory Assignment 1 – Édouard Gagné**

**Question # 1**

I. What is an operating system? What are the main purposes of an operating system?

An operating system is a software. The main purpose of the operating system is to act as an intermediary between the user and the computer hardware.

II. Define the essential properties of the following types of operating systems:

* Batch: Users do not interact with the OS directly and jobs with similar needs are batched together.
* Time sharing: Allows multiple users to use the computer at the same time.
* Dedicated: Designed to be used in specific system.
* Real time: Characterized by a very small response time.
* Multiprogramming: Allows multiple process to be run at the same time on a single processor.

III. Under what circumstances would a user be better of using a time-sharing system rather than a PC or single-user workstation?

If other users also plan to use the system. In this case, time-sharing is a more efficient use of hardware and resources.

III. Under what circumstances would a user be better of using a time-sharing system rather than a PC or single-user workstation?

**Question # 2**

Consider a computer system with a single-core processor. There are two processes to run in the system: P1 and P2. Process P1 has a life cycle as follows: CPU burst time of 15 units, followed by I/O burst time of minimum 10 units, followed by CPU burst time of 10 units. Process P2 has the following life cycle: CPU burst time of 10 units, followed by I/O burst time of minimum 5 units, followed by CPU burst time of 15 units. Now answer the following questions:

a) Considering a single programmed operating system, what is the minimal total time required to complete executions of the two processes? You should explain your answer with a diagram.

Adding all the processing times of both process we botain a total minimal time of 65 units.

b) Now considering a multiprogrammed operating system, what is the minimal total time required to complete executions of the two processes? You should explain your answer with a diagram.

The minimal execution time is the same if either P1 and P2 goes first and is equal to 50 units.

c) Throughput is defined as the number of processes (tasks) completed per unit time. Following this definition, calculate the throughputs for parts a) and b) above. How does multiprogramming affect throughput? Explain your answer.

For a), we have a throughput of 0.03 and for b) we have a throughput of 0.05. It is clear that multiprogramming results in a higher throughput and is therefore more efficient than uniprogramming.

**Question # 3**

I. What is the performance advantage in having device drivers and devices synchronize by means of device interrupts, rather than by polling (i.e., device driver keeps on polling the device to see if a specific event has occurred)? Under what circumstances can polling be advantageous over interrupts?

Interrupts are more efficient than polling when the CPU keeps on polling a device and does not find any device ready for servicing. Polling is more efficient than interrupts when the devices keep interrupting the CPU processing repeatedly.

II. Is it possible to use a DMA controller if the system does not support interrupts? Explain why.

No, because a single interrupt is needed to tell the device driver that the transfer of the block of data through DMA is completed.

III. The procedure ContextSwitch is called whenever there is a switch in context from a running program A to another program B. The procedure is a straightforward assembly language routine that saves and restores registers, and must be atomic. Something disastrous can happen if the routine ContextSwitch is not atomic.

(a) Explain why ContextSwitch must be atomic, possibly with an example.

If a context switch happens in process A and it is not atomic, then process B could interrupt the context switch and use resources that still belong to process A, which can cause lots of problems.

(b) Explain how the atomicity can be achieved in practice.

Disabling interrupts when a context which is occurring would render the context switch atomic.

**Question # 4**

I. If a user program needs to perform I/O, it needs to trap the OS via a system call that transfers control to the kernel. The kernel performs I/O on behalf of the user program. However, systems calls have added overheads, which can slow down the entire system. In that case, why not let user processes perform I/O directly, without going through the kernel?

The user cannot be trusted to make proper use of the I/O. The kernel is needed to ensure that an I/O device is not used by two processes at the same time (where that would cause a problem).

II. Consider a computer running in the user mode. It will switch to the monitor mode whenever an interrupt or trap occurs, jumping to the address determined from the interrupt vector.

(a) A smart, but malicious, user took advantage of a certain serious loophole in the computer's protection mechanism, by which he could make run his own user program in the monitor mode! This can cause disastrous effects. What could have he possibly done to achieve this? What disastrous effects could it cause?

The user probably manually forced an interrupt through an infinite loop, which allowed him to run the rest of his program in monitor mode. This allows him to have unrestricted access to the hardware and memory and cause serious damage to the system.

(b) Suggest a remedy for the loophole

Implementing a handler to handle user generated interrupt and prevent the user to enter monitor mode in this case would fix this problem.

**Question # 5**

Suppose that a multiprogrammed system has a load of N processes with individual execution times of t1, t2, ...,tN. Answer the following questions:

a) How would it be possible that the time to complete the N processes could be as small as: maximum (t1, t2, ...,tN)?

If each process use different devices and resources and are not waiting for another to finish a task then the execution time will be equal to the highest execution time amongst the processes.

b) How would it be possible that the total execution time, T > t1+ t2+ ...+tN? In other words, what would cause the total execution time to exceed the sum of individual process execution times?

If a device or resource is locked by a process and other processes are waiting on it to begin executing. For example, if process 1 is using a device that process 2 needs to start, then process 2 will have to wait until process 1 is done to execute. With overhead, this will cause the total process time to be greater than the total processing time of the processes.

**Question # 6**

Which of the following instructions should be privileged? Explain why.

(i) Read the system clock: Unprivileged, every process should be able to read the clock.

(ii) Clear memory: Privileged, since a process could wipe other processes’ memory.

(iii) Reading from user space: Unprivileged, processes should be able to read from their own user space without kernel mode

(iv) Writing to user space: Unprivileged, same as above.

(v) Copy from one register to another: Privileged, a process shouldn’t be able to access registers used by other processes.

(vi) Turn off interrupts: Privileged, the user should not be able to turn off interrupt and crash the system using an exploit like an infinite loop.

(vii) Switch from user to monitor mode: Privileged, the user should not be able to switch to kernel mode anytime they want for security reasons.

**Question # 7**

Assume you are given the responsibility to design two OS systems, a Network Operating System and a Distributed Operating System. Indicate the primary differences between these two systems. Additionally, you need to indicate if there any possible common routines between these systems? If yes, indicate some of these routines. If no, explain why common routines between these two particular systems do not make sense.

A Distributed system allows resource sharing by systems connected through a network. A network operating system is a specialized OS that allows shared files and printers through computers in a network. It is then obvious that a distributed operating system will have some routine common to the network operating system, since the resources that are shared through the distributed OS include network related ressources.