Mohammad Ali Zahir

ID: 40077619

COMP 346: Theory Assignment 1

1.

i. An operating system is process which creates a liaison of hardware between users and applications. An operating system acts as an ease of use and improved performance for the computer.

ii. Batch: needed for efficiency for multiprogramming

Timesharing: creates interactive computing so users can interact with multiple jobs

Dedicated: used for a single purpose program without the need of switching to another program

Real-time: processes data in real-time without the use of a buffer in the computer memory

Multiprogramming: organizes jobs so CPU only has one to execute

iii. Time sharing would make sense when there is more than one user who use the machine, but it is a huge task. Since the number of users would be limited, the power of the one task can be shared by the hardware, and it would not be harmful to the overall performance. For a small task, a personal computer is best since it would suit to a user's own needs, and it does not require multiple users to work on such a task.

2.

a) In terms of the values that we have already

Process/Time	CPU	I/O	CPU
P1	15	10	10
P2	10	5	15

For single programmed, we would have to traverse through all the processes, hence it would look like

15	**10**	**10**	**10**	**5**	**15**

If we calculate all these together, the total time **would be equal to 65 units**.

b) For the multi-programmed, one process would run while the other process would wait for I/O, hence this would mean that we would skip the I/O time in the calculation completely.

Hence the processes would look like

15	**10**	**10**	**15**

If we calculate all these together, the total time **would be equal to 50 units**.

c) Throughput is calculated by the number the number of processes (tasks) completed per unit time.

Based on this,

Single Programmed: Time/ Number of processes

65/2 = 32.5

Multi-programmed: Time/Number of processes

50/2 = 25

This would mean the throughput of the multi-programmed OS allows it to perform more tasks and reduce CPU time.

3.

i. Advantages of having interrupt over polling is that whenever the CPU requires it is attention, it will interrupt the current process to give attention to it. This allows the overall hardware to not waste any CPU cycles. On the other hand, polling will constantly check the status of the register to see if the event has occurred. This would mean a lot of CPU cycles would be wasted with polling. However, if the jobs on the I/O are short and frequent, polling would be better suited since it would not waste CPU cycles.

ii. With polling, it is possible to use the DMA controller. With this we can check the change of status in a particular device since it would access every memory bit one by one.

iii.

a) If the process is not atomic, we would have inconsistencies in the results overall. For example, the registration at a gym. If context switching happens

in the middle of the process, we might be registered with the wrong phone number and would then have inconsistencies.

b) To achieve atomicity, we could synchronise the methods. This would ensure that no inconsistencies would happen one process will only work if the other is produced.

4.

I. As system codes are stored in the memory area in kernel mode, I/O needs to be executed by the use of these system codes. We can switch to user mode to get access to these codes, however, we will then render them vulnerable to an attack, hence I/O can only be done in kernel mode.

II.

a) When inserting a malicious code, we would have to account for the memory space it would take in kernel mode. When we go into the interrupt, the code can either be in kernel or user mode. If it goes to kernel mode, it will be read only which would mean that it would not be as harmful as it can't change the memory directly. When it is in user mode, it can directly point to the memory in the kernel which then he could change the position of the interrupt, hence allowing the code to function.

b) One solution would be to privilege the position of the interrupt and to ban the user to modify its location as well. This would not allow the code to do damage in the CPU.

5.

a) If we let N be the number of processes. If we execute the process one less than the maximum (which in this case would be N – 1 times), we could achieve maximum time if it doesn't increase the waiting time.

b) If the waiting time of one process overlaps with another, this would make it that the total execution time would increase. As in a single programmed, only one process can run at a time, and since one process would need to end before the other would need to run, this increases the waiting time, which would hence increase the execution time.

6.

i. Since you have no control over the time that you read**, this would not need to be privileged.**

ii. As you don't want memory to be just cleared randomly, **this would need to be privileged.**

iii. Since it is carried in user mode, **this would not need to be privileged.**

iv. Once again, since this is also carried out in user mode, **this would not need to be privileged.**

v. As you copy one register to another, there might be a chance that you enter in kernel mode, which would mean that there is a possibility for this instruction to be critically fatal, **hence this would need to be privileged.**

vi. Switching off an interrupt would mean that the process would reach kernel mode, and this could use the whole CPU, **hence this would need to be privileged.**

vii. As this is a normal system call, **this would not need to be privileged.**

Hence, ii, v, and vi need to privileged.

7.

Network Operating System: - Offers communication to machines to a network

- Allow messages to be exchanged between these machines, this allows the whole network of machines to act as one single unit.

Distributed Operating System: -

Allows for different communication paths, which allow for connection to other different websites.

-Mostly used by computer users due to it's ability to multitask

- Allow resources stored in a single hardware to be shared amongst participating users by shared memory.

Aside from the fact that they both use communication path to share data, there is nothing else common between these two.