COMP 346 – Fall 2018
Theory Assignment 1

Due Date & Submission Format: *September 28, 2018.*

Name: Tianlin Yang
ID: 40010303

Question # 1

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

Answer:
An operating system is software that manages a computer's hardware. It also provides a basis for application programs and acts as an intermediary between the computer user and the computer hardware.

Main purpose:
1. Provides an environment with in which other programs can do useful work.
2. Make the computer ease of use.
3. Controls the hardware and coordinates its use among the various application programs for the various users. [1]

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

- Batch
- Time sharing
- Dedicated
- Real time
- Multiprogramming

Answer:
Batch:
In computing, batch processing refers to a computer working through a queue or batch of separate jobs without manual intervention. A batch operating system is an operating system in which the same type of processes is batched together for execution. It's a relatively faster system than a traditional system. Batch systems processed jobs in bulk, with predetermined input from files or other data sources. The users of a batch operating system do not interact with the computer directly. [2]

Time sharing:
Time-sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time. Processor's time which is shared among multiple users simultaneously is termed as time-sharing. Time-sharing systems used a timer and scheduling algorithms to cycle processes rapidly through the CPU, giving each user a share of the resources. [3]

Dedicated:
A dedicated system is a computer system capable of performing one specific task.

Real time:
A real-time operating system (RTOS) is an operating system (OS) intended to serve real-time applications that process data as it comes in, typically without buffer delays. [4]

Virtualization:
In computing, virtualization refers to the act of creating a virtual (rather than actual) version of something, including virtual computer hardware platforms, storage devices, and computer network resources. [5]

DMA:
DMA (direct-memory-access) is a feature of computer systems that allows certain hardware subsystems to access main memory, independent of the CPU. Many computers avoid burdening the main CPU with PIO by offloading some of this work to a DMA controller. [6]

Interrupt Timeline:
Interrupt Timeline is a graph that a device sends the request and transfer data after the transfer done, CPU goes to handle device interrupt processing.

Daisy chain:
A daisy chain is a wiring scheme in which multiple devices are wired together in sequence or in a ring. When device A has a cable that plugs into device B, and device B has a cable that plugs into device C, and device C plugs into a port on the computer, this arrangement is called a daisy chain. A daisy chain usually operates as a bus. [7]

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

Answer:

First, user can use remote control to access time-sharing system do the work without a PC or single-user workstation.

Second, if large number of users need access the resources at the same time, the TSS is better.

Third, for FEA simulation and calculation which will take long time (around a week per test on PC), TSS is better because its faster.

Question # 2

Consider a computer system with a single-core processor. There are two processes to run in the system: P_1 and P_2. Process P_1 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 P_2 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.

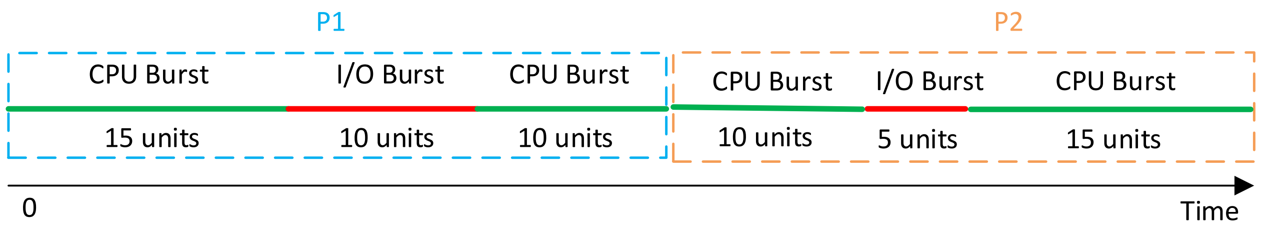
b) Now considering a *multiprogrammed* operating system, what is the minimal total time required to complete executions of the two processes? You should explain your answer with a diagram.

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.

Answer:

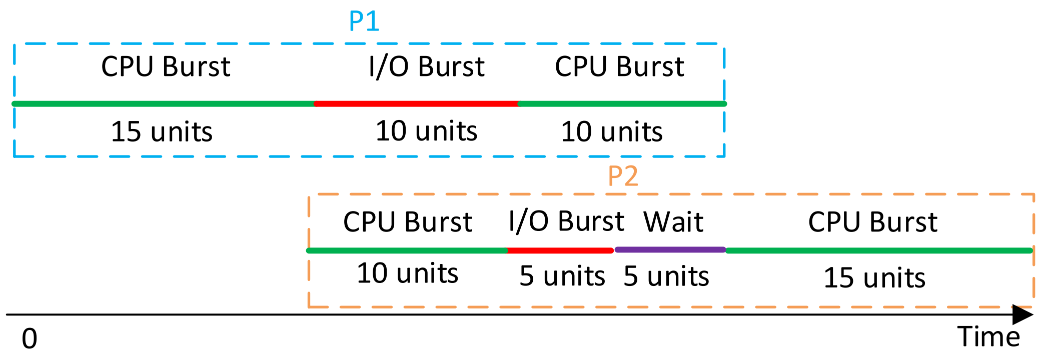
(a) In a system with a single CPU core, only one process can run at a time. Others must wait until the CPU's core is free and can be rescheduled. In a simple computer system, the CPU then just sits idle. All this waiting time is wasted; no useful work is accomplished. [8]

Diagram:



Minimal completion time $(15 + 10 + 10) + (10 + 5 + 15) = 65\ units$

(b) With multiprogramming, we try to use this time productively. Several processes are kept in memory at one time. When one process has to wait, the operating system takes the CPU away from that process and gives the CPU to another process.
Diagram:



Minimum completion time (green color) $15 + 10 + 10 + 15 = 50\ units$
If P2 runs first, the same total execution time will be obtained.

(c) Throughput for part (a) = 2/65 = 0.0308 (tasks/unit time)
Throughput for part (b) = 2/50 = 0.04 (tasks/unit time)

Compare with single programed system, multiprogramming can utilize idle time of CPU. When one process transfers to I/O burst condition, CPU will transfer to another process. Therefore, it keeps the CPU as busy as possible and make the throughput larger.

<u>Question # 3</u>

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?

Answer:

Devices running speed often lower than the CPU. Therefore, when use polling, the CPU has to waiting the devices response frequently. Interrupt can improve the efficiency of CPU and be used for multitasking.

In other case, if the device speed is very fast which similar to the CPU, Or the frequently events occurs without urgent, the polling option better than interrupts.

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

Answer:

No, Although the DMA controller proceeds to operate the memory bus directly, placing addresses on the bus to perform transfers without the help of the main CPU. When the devices finish their tasks, the device still need to interrupt CPU to update the results and statues. So the interruption is necessary.

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

Answer:

(a) When a context switch occurs, the system needs to save the current context (process state and values of CPU registers) of the running process so that it can resume execution the next time it runs from exactly where it left. Saving of context must be atomic to ensure that the context of the process will be stored completely without creating any inconsistency. Similarly restoring the context of a ready to run process must also be atomic to ensure its correct execution. Note that context switch code executes inside the kernel, and critical section of this code manipulates shared kernel variables and data-structures; hence atomic execution of these critical sections is essential for consistency of shared data.[9]

Failure to ensure atomicity of critical sections could result in inconsistencies, for instance: a process executing with some of the registers containing values from another process, or the program counter with an erroneous value, or the ready queue pointers in an inconsistent state, etc. This would result in the process running in an unforeseen way, which could lead to its termination or damage to the system or other processes.

(b) Atomicity can be achieved by disabling interrupts at the beginning of a context switch and enabling interrupts after completing the context switch. This will guarantee that context switch occurs without interference from any other operations, and the context of the previous process is saved completely and the context of the new process is loaded exactly from the state where it left the CPU. Moreover, it will ensure consistency of the kernel data structures (e.g., ready queue, device queues, etc.) and registers which are manipulated during a context switch. Note that disabling and enabling of interrupts is done inside the kernel executing in the kernel mode and hence it is necessarily safe.[10]

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?

Answer:

User mode and kernel mode have different permission. In Kernel mode, the executing code has completed and unrestricted access to the underlying hardware. If a user program can processes perform I/O directly, it means user program can have same permissions as kernel and any mistakes may cause system broken or even hardware damages. OS is designed to protect hardware from some malicious program or wrong operation.[11]

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?

Answer:

When a user program made system calls, the computer found tap instructions numbers. And it changed to kernel mode. Also, the malicious user changed the system call address into the address of user program. Therefore, the user program can be executed in kernel mode. The program can access any address directly and may cause system broken or even hardware damages.[12]

(b) Suggest a remedy for the loophole.

Answer:

When the CPU execute the system program in the system call address, a protect layer should be established for checking the address reliability in order to find imitative.

Question # 5

Suppose that a multiprogrammed system has a load of N processes with individual execution times of $t_1, t_2, \ldots, t_N$. Answer the following questions:

a) How would it be possible that the time to complete the N processes could be as small as: *maximum* $(t_1, t_2, \ldots, t_N)$?

Answer:

Consider one of the N processes as Pm, which has the maximum execution time. If we could execute all the remaining N-1 processes during the waiting times of Pm, and if none of them increases the waiting time of Pm, then the completion time for N processes will be equal to maximum (t1, t2,…,tN). For a multiprocessor system, if there are at least N processors, then all the N processes could run in parallel. Then, without any other overheads, this could also give execution time equal to maximum (t1, t2,…,tN).[13]

b) How would it be possible that the total execution time, $T > t_1 + t_2 + \ldots + t_N$? In other words, what would cause the total execution time to exceed the sum of individual process execution times?

Answer:

Total execution time could be greater than the sum of all process execution times if it not possible to overlap the waiting time of the one process with the execution times of the other processes (e.g., processes are waiting for timer events like clock interrupt) which make all processes wait simultaneously. This is compounded with context switch overhead. Another situation is when all the processes are CPU-bound and they do not block for I/O execution. So, the context switch overhead makes the total execution time greater than the sum of all process execution times. [14]

Question # 6

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

(i) Read the system clock,
(ii) Clear memory,
(iii) Reading from user space
(iv) Writing to user space
(v) Copy from one register to another
(vi) Turn off interrupts, and
(vii) Switch from user to monitor mode.

Answer:

1. Read the system clock

Not privileged. Reading the clock can't interfere with the kernel or other user programs.

2. Clear memory.

Privileged. This would wipe out the code and data of the OS and any other processes. User level processes should not be allowed to do this.

3. Reading from user space

Not privileged. Reading the user space can't interfere with the kernel or other user programs.

4. Writing to user space

Not privileged. Reading the user space can't interfere with the kernel or other user programs.

5. Copy from one register to another

Not privileged. There are special purpose registers which require privileged instructions, but general registers do not require privileged instructions.

6. Turn off interrupts.

Privileged. This would allow a denial of service attack on the OS and the other user processes. Only the OS should be in charge of setting the flow of control between the kernel and the user processes (except in the case of voluntary yielding).

7. Switch from user to monitor mode.

Privileged. If user programs could switch to monitor mode, they could do whatever they wanted. Instead, they must request a switch to monitor mode and then yield control to the kernel by making a system call.

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.

Answer:

Differences:

A network operating system acts autonomously from all other computers on the network, although it is aware of the network and is able to communicate with other networked computers. A distributed operating system provides a less autonomous and users are not aware of multiplicity of machines.[15]

Common routines:

Network and Distributed Operating systems have a common hardware base, Kernel, Management and Scheduler.

REFERENCE

[1] Abraham S, Peter B.G, Gear G, Operating System Concepts 10th , John Wiley & Sons, Inc. ISBN 978-1-118-06333-0, Page 3-4.
[2] Abraham S, Peter B.G, Gear G, Operating System Concepts 10th , John Wiley & Sons, Inc. ISBN 978-1-118-06333-0, Page 69. And Wikipedia: https://en.wikipedia.org/wiki/Batch_processing
[3] https://www.quora.com/What-is-time-sharing-operating-system-with-example
[4] https://en.wikipedia.org/wiki/Real-time_operating_system
[5] https://en.wikipedia.org/wiki/Virtualization
[6] https://en.wikipedia.org/wiki/Direct_memory_access
[7] https://en.wikipedia.org/wiki/Daisy_chain_(electrical_engineering)
[8] Abraham S, Peter B.G, Gear G, Operating System Concepts 10th , John Wiley & Sons, Inc. ISBN 978-1-118-06333-0, Page 200.
[9] https://www.cs.ubc.ca/~tmm/courses/213-12F/slides/213-2b-4x4.pdf
[10] http://bnrg.cs.berkeley.edu/~adj/cs16x/exams/fa05mt1-solutions.pdf
[11] https://blog.codinghorror.com/understanding-user-and-kernel-mode/
[12] https://www.webopedia.com/DidYouKnow/Internet/virus.asp
[13] https://social.microsoft.com/Forums/en-US/fc0cbd0a-4355-4338-ace8-9ce933d64a1d/multiprogrammed-system?forum=whatforum
[14] https://cgi.cse.unsw.edu.au/~cs3231/07s1/example_exam_answers_updated.html
[15] http://www.padakuu.com/article/181-distributed-system and https://www.quora.com/What-is-the-difference-between-Network-OS-and-Distributed-OS