Assignment #1

Tuesday, March 4, 2025 10:34 AM

Chapter 1 (30pts, 5pts each)

1. What is a monolithic kernel and a micro kernel? Please compare the pros and cons of them.

A monolithic kernel is a commonly used kernel, especially in Linux. It is a large process running entirely in the kernel address space (a single binary file). All the kernel services execute in the kernel address space.

A microkernel is a kernel broken down into separate processes (aka servers) that are kept separate and run in different address spaces. Communication is done via message passing (higher overhead) through Inter-Process Communications (IPC).

Pros:
Monolithic Kernel is very fast
Microkernel is modular, extensible, easily maintained, more reliable, and secure.

Cons:
A monolithic Kernel is huge and more challenging to maintain. It does not protect kernel components, has complex dependencies among components, and it is not very extensible.
Microkernel has performance loss and has complicated process management.

2. Please explain the three-stage process of hard disk read/write.

A hard disk (HDD) read/write process has three stages. They are the following:

1) Seek time - this has to do with the time it takes to move the arm over the hard disk to the proper tack
2) Rotational latency - this is when the arm is over the proper track, but is waiting for the desired sector to rotate under the arm's read/wrtie head.
3) Transfer time - this is the time it takes to transfer a block of bits (sector) from under the read/write head.

3. What are the differences between a system call and an I/O interrupt? Please briefly explain.

A system call (aka Trap) is when a user program (from the user space) requests services from the Operating System (the kernel space).
An I/O interrupt is the suspension of a process caused by an external event performed so that the process can be resumed later. This improves processing efficiency.

4. Describe the steps an OS must take to process an I/O interrupt.

1) The CPU issues commands to the I/O device by writing parameters (e.g., disk address, read/write mode) into the device's registers.
2) The I/O device completes its operation and signals the interrupt controller.
3) The interrupt controller notifies the CPU about the interrupt event.
4) The CPU saves its current process state, switches to kernel mode, and locates and invokes the appropriate interrupt handler.
5) The interrupt handler processes the event (e.g., reads data, clears the interrupt), after which the CPU restores the saved state and resumes the interrupted process.

5. Explain briefly the three ways to transition from the user mode to kernel mode.

1) Hardware Interrupts - A hardware device triggers an interrupt that causes the CPU to switch into kernel mode.
2) System Calls (Traps) - A user program deliberately invokes a trap instruction, transferring control to kernel mode.
3) Exceptions - An error or unexpected user code condition automatically triggers a kernel mode switch.

6. How can I/O devices notify OS of I/O events? Please list and explain three different ways.

1) Polling - The OS periodically checks a status register where the I/O device stores event information. This method is simple but can waste CPU cycles and increase latency.
2) Interrupts - The I/O device sends a signal to the CPU when it needs attention, prompting an immediate switch to kernel mode to handle the event. This is more responsive than polling, though more complex.
3) DMA - The device directly transfers data to or from memory without continuous CPU involvement, reducing CPU overhead during large data transfers.

Chapter 2 (20pts)

1. What is multi-programming? Please explain why it is needed (5 pts).

Multi-programming refers to having multiple processes (or programs) loaded in memory at once so the CPU can switch to a ready process whenever another is waiting for I/O. It is needed because it prevents the CPU from sitting idle during I/O waits, boosting overall system efficiency and throughput.

2. Assume that the following processes are to be executed on a uniprocessor system. Please calculate the average turnaround time and response time of these processes under the following scheduling policies. Process switch overhead, interrupts, and blocking system calls are ignored (10 pts in total).

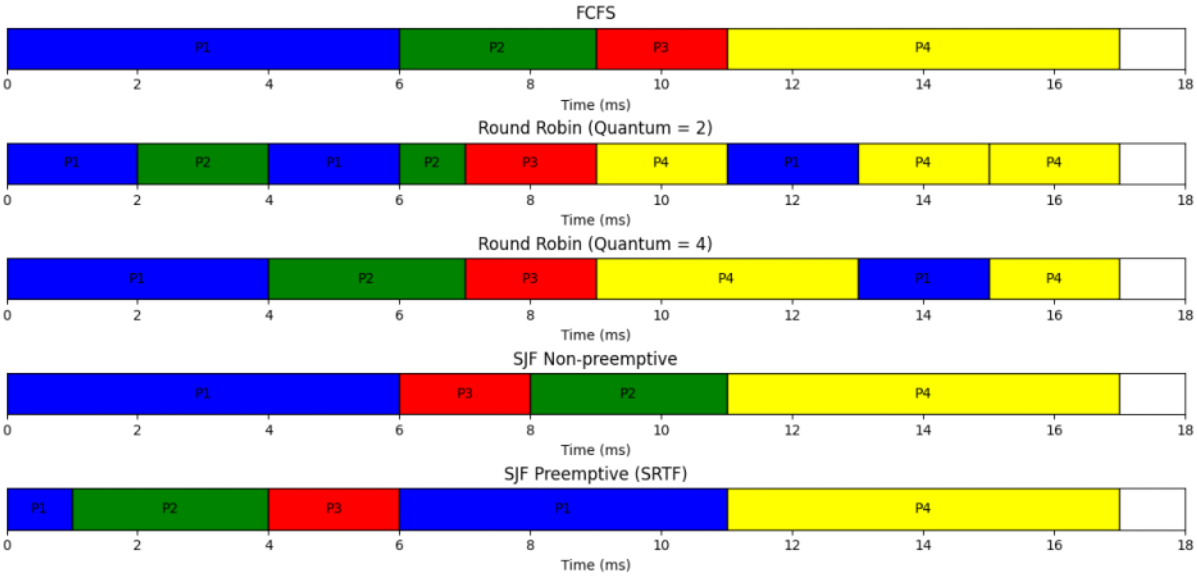
(a) FCFS (2pts)

(b) Round Robin (quantum = 2 and 4, 4pts)

(c) Shortest Job First (both preemptive and non-preemptive, 4pts)

Process	Process Arrival Time	CPU burst
P1	0	6
P2	1	3
P3	3	2
P4	4	6

Response time = first response time – arrival time
Turnaround time = finish time – arrival time



(a) First Come First Served

Average turnaround time = ((6-0) + (9-1) + (11-3) + (17-4)) / 4 = 8.75
Average response time = ((0-0) + (6-1) + (9-3) + (11-4)) / 4 = 4.5

(b) Round Robin quantum = 2

Average turnaround time = ((13-0) + (9-1) + (8-3) + (17-4)) / 4 = 9.75
Average response time = ((0-0) + (2-1) + (6-3) + (9-4)) / 4 = 2.25

(b) Round Robin quantum = 4

Average turnaround time = ((15-0) + (7-1) + (9-3) + (17-4)) / 4 = 10
Average response time = ((0-0) + (4-1) + (7-3) + (9-4)) / 4 = 3

(c) Shortest Job First (SJF) preemptive

Average turnaround time = ((11-0) + (4-1) + (6-3) + (17-4)) / 4 = 7.5

Average response time = ((0-0) + (1-1) + (4-3) + (11-4)) / 4 = 2.0

(c) Shortest Job First (SJF) non-preemptive

Average turnaround time = ((6-0) + (8-3) + (11-1) + (17-4)) / 4 = 8.5
Average response time = ((0-0) + (6-3) + (8-1) + (11-4)) / 4 = 4.25

3. A single-CPU real-time system has four periodic events with periods of 50, 100, 200, and 200ms each. Suppose that the four events require 30, 20, 10, and x ms of CPU time, respectively. What is the largest value of x for which the system is schedulable? Context switch overhead etc. is ignored. (5 pts)

Schedulable real-time system

- Given
  - *m* periodic events
  - event *i* occurs within period P_i seconds and requires C_i seconds
- Then the load can only be handled (schedulable) if

$$\sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1$$

Using this formula, we have the following:

30/50 = 0.6
20/100 = 0.2
10/200 = 0.05
and x/200

The summation of these are as follows:

0.6 + 0.2 + 0.05 +x/200 <= 1
0.85 + x/200 <= 1
so x/200 <= 0.15
which means x must be <= 30
So, the largest value of "x" for which the system is schedulable is 30 ms.