# IDATA2305 Operating Systems with System Programming

## This document includes questions and answers for the IDATA2305 course, covering topics that may appear in exams.

## Chapter 1 - Introduction

**Why is an Operating system required? What are its functions?**

An operating system is required for managing computer hardware resources efficiently. Some of its functions are process management, memory management, file system management, device management, security networking and error handling.

**How does a device controller inform the CPU that it has finished its operation?**

A device controller typically uses an interrupt mechanism to inform the CPU that it has finished its operation. When a device completes its task, it triggers an interrupt signal to the CPU, causing the CPU to temporarily pause its current operation and handle the request from the device controller. The CPU then executes an interrupt service routine (ISR) to process the completed operation and resume normal execution afterward. This mechanism allows the CPU to efficiently manage multiple devices and respond promptly to their requests without wasting resources by continuously polling them for completion.

**How does the CPU move its data regarding local buffers?**

The CPU moves data from memory to local buffers and vice versa. These local buffers can include various types of cache memory, such as L1, L2 and L3 cache. These are all located closer to the CPU than the main memory (RAM).

**The program must be in which memory to be executed directly by the CPU?**

For a program to be executed directly by the CPU, it must be loaded into the main memory (RAM). The CPU cannot use cache memories. Though these memories are a lot faster than the RAM, they are much smaller in size and cannot fit entire programs. However, the cache will be used during the execution of the program.

**Why does the CPU execute data from main memory and not secondary storage?**

In contrast to cache memories, which have too small capacity, why can the CPU not just execute data from a secondary storage? The main factor is speed. Secondary storage (Which is hard disks, solid state drives, etc.) is not nearly as fast as RAM and would significantly slow down access times. The CPU also does not have direct access to secondary storage through the memory bus.

**What is an interrupt request?**

An “interrupt” is a signal sent by a hardware device or software to the CPU, indicating that it requires attention. For Example:A user clicks the mouse to run a program, and a request is sent to the CPU to suspend the currently running program so that the CPU can run the new program given by the mouse click.

**Describe all these terms with an example for each.**

***Interrupt handling routine.***

An interrupt handling routine is a piece of code that is executed by the CPU in response to an interrupt signal from hardware or software.  
Example: See previous.

***Application program***

An application program is a software designed to perform specific tasks or provide specific functionality for end-users.  
Example: Microsoft Word, Web browsers, etc.

***System program***

System programs are software programs that provide essential system-related functionality and services to both users and other software applications.

Example: Disk defragmenter utility is an example of a system program. It

***Kernel***

The kernel is the core component of an operating system that manages system resources, such as memory, processes, and hardware devices.

Example: The Linux kernel.

***Operating system***

An operating system is system software that manages computer hardware, software resources, and provides common services for computer programs.  
Example: Windows, macOS, Linux Mint.

**What is the term for the communication channels through which the CPU and Input/Output (I/O) device controllers exchange data?**

The communication channels are called buses.

**What are input and output devices?**

Input and output (I/O) devices are peripheral devices that enable communication between a computer system and the external world. They facilitate the input of data into the computer and the output of processed information from the computer.

**Are monitors and printers input devices or output devices?**

Yes, Monitors and printers are I/O devices as they serve output functionality.

**Is a webcam an input device or output device?**

Yes, a webcam is an I/O device as it serves as an input device.

**Why is caching memory advantageous? Where is it located?**

Caching memory is advantageous because:

1. Faster Access: Cache memory indeed offers faster access times compared to main memory and secondary storage devices. This rapid access speed helps in speeding up data retrieval and instruction execution, thereby enhancing overall system performance.
2. Reduced Latency: By storing frequently used data and instructions, cache memory reduces the latency associated with fetching data from slower main memory or secondary storage. This means that the CPU can quickly access commonly used data without experiencing significant delays, leading to smoother and more efficient operation.
3. Proximity to CPU: Cache memory is strategically located closer to the CPU than other types of memory on the motherboard. This proximity ensures that cache memory can be accessed with minimal delay, maximizing its effectiveness in speeding up data retrieval and processing.

**Identify volatile and non-volatile memory? Also, which memory is the slowest?**

Volatile memory will lose its data when power is cut, while non-volatile will not.

a) Magnetic tape: Non-volatile. This is also much slower.

b) Random Access Memory: Volatile.

**A java code expects values less than 10 to be entered by the end user. A user enters 12 as input. What is this error in operating system terminology? Is the error software generated or hardware generated?**

An error caused by this scenario is called a runtime error. This is a software-generated error because it arises from the logic or behavior of the software program itself.

**Why do we need a multitasking operating system? How does it work?**

We need a multitasking operating system to allow multiple programs to run concurrently on a single computer, enabling better resource utilization. It works by allocating CPU time to each program in small increments, rapidly switching between them so it appears that they are running simultaneously.

**What is multiprogramming, and how is it different from multitasking?**  
Multiprogramming involves running multiple programs simultaneously by sharing a single processor. Multitasking refers to the ability of an OS to switch between different programs rapidly, giving the illusion of simultaneous execution. The key difference is that multiprogramming focuses on maximizing CPU utilization, while multitasking focuses on user interaction and responsiveness.

**What is the difference between process management and memory management?**

Process management focuses on managing processes within the operating system, including creation, scheduling, synchronization, and communication, while memory management involves allocating and deallocating memory resources efficiently among processes, ensuring memory protection, and managing memory accesses.

**How are interrupts and exceptions different?**

Interrupts are caused by hardware failures, while exceptions are caused by software errors.

**What are advantages of using base and limit registers?**Base and limit registers provide memory protection and isolation. It prevents unauthorized processes from accessing memory outside their allocated regions.

## Chapter 2 – OS internals

**What are some OS interfaces, and how do they work?**

* A CLI, or Command Line Interpreter allows direct command entry. It is sometimes implemented in a kernel, or by a system program. There are also sometimes multiple flavors implemented – shells. The main task of the CLI is to fetch a command from a user and execute it.
* Examples of other interfaces include Touchscreen, desktop GUI.

**What is Standard API?**

Standard API (Application Programming Interface) refers to a set of standardized functions and protocols provided by the operating system to facilitate interactions between software applications and the underlying hardware and system resources. Standard API code is executed in user mode.

**What is API – System Call – OS Relationship?**  
APIs serve as interfaces to interact with the operating system. When applications make requests through APIs, they often trigger system calls, which are requests from user-level processes to the operating system kernel for specific services or resources. System Calls are executed in kernel mode.

**Name all six types of System Calls**- Process control  
- File management  
- Device management  
- Information maintenance  
- Protection and security.

**What are Linkers and Loaders?**  
In short, they are components that converts source code into executable programs.

## Chapter 3 – Threads and Concurrency

**Identify the basic components of a thread.**

* Thread Control Block: Contains info about threads execution state.
* Thread Identifier: Unique identifier assigned to each thread.
* Program Counter: Points to the next instruction to be executed within the thread.
* Stack: Each thread has its own stack, which stores local variables, function parameters, and return addresses.
* Thread State: Represents the current state of the thread.

**What is the difference between threads and processes?**Threads are lightweight units of execution within a process, sharing the same memory space and resources, making communication and data sharing efficient. Processes, on the other hand, are independent units of execution with their own memory space and resources, requiring inter-process communication mechanisms for communication and data sharing.

**Describe the 4 main benefits and challenges of designing multithreaded applications.**

* Responsiveness: May allow continued execution if part of process is blocked, especially important for user interfaces.
* Resource sharing: Threads share resources of process, easier than shared memory or message passing.
* Economy: Cheaper than process creation, thread switching lower overhead than context switching.
* Scalability: Process can take advantage of multicore architectures.

## Chapter 4 – Thread Synchronization

**What are Race Conditions in threads?**Race conditions in threads are situations where two or more threads access the same variable or data in a way where the result stored in the variable depends on how threads access the variable is scheduled.

**How can we prevent Race Conditions?**To prevent race conditions from occurring you must make sure that the critical section is executed as an atomic instruction. That means that once a single thread is executing it, no other threads can execute it until the first thread has left the critical section. In Java it can be achieved by adding “synchronized” to the method like this:

“public synchronized void myMethod()”

**What is Thread Starvation?**  
Thread starvation happens when a thread is unable to access the CPU or resources it needs, often due to scheduling prioritization or resource contentions. To prevent this, it is important to focus on fairness. Fairness in thread scheduling aims to ensure equal treatment of threads in resource allocation and CPU time to prevent starvation.

## Chapter 5 – CPU Scheduling

**Why is CPU Scheduling so important?**  
CPU scheduling is important because it determines how effectively a system utilizes its CPU resources. By allocating CPU time to processes or threads, scheduling algorithms impact system performance, responsiveness, and efficiency. Effective CPU scheduling ensures optimal resource utilization, minimizes response times, and enhances overall system throughput, contributing to a smooth and responsive user experience.

**How does multiprogramming contribute to maximizing CPU utilization, and what are the key aspects of the CPU-I/O burst cycle in process execution?**Multiprogramming maximizes CPU utilization by allowing multiple processes to share the CPU. Process execution involves alternating CPU and I/O bursts, with I/O bursts occurring when processes wait for data to be read from or written to external storage devices. The distribution of CPU bursts is a significant consideration in system performance.

**What is Preemtive and non-Preemtive scheduling?**

* Preemtive scheduling: The running process can be interrupted and must release the CPU.
* Non-Preemtive scheduling: The running process keeps the CPU until it voluntarily releases the CPU.

**Explain the terms regarding Scheduling Criteria**

* CPU utilization – keep the CPU as busy as possible.
* Throughput – number of processes that complete their execution per time unit.
* Turnaround time – Interval from time of submission of a process to the time of completion of the same process.
* Waiting time – amount of time a process has been waiting in the ready queue.

**Explain the main CPU scheduling algorithms**  
First Come First Serve (FCFS): Process that arrive first, will be executed first, no matter the burst time.

Shortest Job First (SJF): When CPU is available it is assigned the process with the shortest execution time when it became available.

Priority Scheduling: When CPU is available it is assigned the process with the highest priority value.  
  
Round Robin (RR): CPU time is allocated in fixed slices, allowing each process to execute for a predefined interval before switching to the next.

## Chapter 6 – Deadlocks

**Define the four necessary conditions that characterize deadlock.**

*Mutual Exclusion*: Only one process can access a resource at a time.

*Hold and Wait*: Processes can request additional resources while holding onto the ones already allocated.

*No Preemption*: Resources cannot be forcibly taken away from a process.

*Circular wait*: A circular chain of processes exists, where each process holds a resource that is requested by the next process in the chain.

**What is a Resource-Allocation Graph (RAG)?**A Resource-Allocation Graph is a visual representation used in OS to depict resource allocation and resource request relationships among processes.

**Does cycles always lead to deadlock?**  
Cycles in a resource allocation graph indicate a scenario where processes are waiting for resources that are held by other processes in the cycle. While cycles create a potential for deadlock, not all cycles necessarily lead to a deadlock.

**How can we prevent deadlocks?**Deadlocks can be prevented by avoiding one or more of the four necessary conditions mentioned above.

**What is Banker’s algorithm?**The Banker’s algorithm is a deadlock avoidance algorithm used to allocate resources to processes in a safe manner, ensuring that deadlock does not occur. It works by dynamically analysing resources requests.

**How can we recover from deadlocks?**Deadlocks can be recovered from by employing techniques such as process termination, resource preemption, or rollback to a safe state.

## Chapter 7 – Memory Management

**What are some various ways to manage memory?**Some various ways to manage memory involves techniques such as contiguous memory allocation, paging segmentation, virtual memory, memory protection and demand paging.

**Mention the most common memory management algorithms.**

* First fit: Allocates the first memory block that is large enough to accommodate the process.
* Best fit: Allocates the smallest memory block that is large enough to accommodate the process.
* Worst fit: Allocates the largest available memory block, leaving behind the largest unused memory fragment.
* Next fit: Similar to First fit but begins searching for a suitable memory block from the location where the last allocation ended.

**What are some Paging strategies?**

* Demand Paging
* Prepaging
* Page size selection
* Page Replacement

**Shortly explain Basic memory hardware**Basic memory hardware includes RAM for storing actively used data and instructions, ROM for essential firmware, cache memory for frequently accessed data, memory controller for managing dataflow and a memory bus.

**What are logical and physical address spaces?**

Logical address space refers to the range of addresses that a process can use to access memory, as perceived by the process itself. It does not necessarily reflect the actual physical location of the memory. The logical address space is typically divided into fixed-size units called pages or segments.

Physical address space refers to the actual hardware addresses of the memory. It represents the real locations of data in the physical memory chips.

Physical and logical address spaces are bridged by the OS using virtual memory management techniques.

**What is Memory-Management Unit (MMU)?**Memory-Management Unit is a hardware component in a computer that translates logical addresses generated by the CPU into physical addresses in the computer’s memory. It enables the system to utilize virtual memory by managing the mapping between logical and physical addresses.

**What is contiguous and non-contiguous memory?**Contiguous memory means that all parts of a program are stored in one block next to each other. Non-contiguous memory means that parts of the program are scattered in different places.

**What is External and Internal Fragmentation and how can we handle it?**External Fragmentation occurs when free memory blocks are scattered throughout the memory space, making it challenging to allocate contiguous blocks of memory to process even though the total free memory is sufficient. To handle external fragmentation, techniques like compaction or memory allocation algorithms that minimize fragmentation can be employed.

Internal fragmentation happens when memory blocks allocated to processes are larger than required, leading to wasted memory within those blocks. Internal fragmentation can be reduced by using memory allocation strategies that allocate precisely according to the needs of the process, such as dynamic memory allocation (First fit, best fit, etc).

## Chapter 8 – Virtual Memory Management

**What is Virtual Memory and why do we need it?**Virtual Memory is a memory management technique used by OS to provide the illusion of a larger memory space that is physically available in the system’s RAM. It works by using a combination of hardware and software to temporarily transfer data from RAM to disk storage, creating a virtual memory space.

**What is Demand Paging?**Demand Paging is a memory management technique that optimizes memory usage by loading pages into memory only when needed, reducing initial loading time and conserving memory resources. However, it also introduces performance overhead due to page faults, which must be managed effectively by the OS.

**What is a Page Fault and how can we handle one?**A page fault occurs when a program accesses a memory page that is not currently in the RAM. To handle a page fault, the OS fetches the required page from the disk into the RAM, updating the page table accordingly. If there is no free memory, the OS selects a page to evict using a page replacement algorithm.

**What is Page Replacement?**Page Replacement is a process in memory management where the OS selects a page in RAM to be removed when there is a need to bring in a new page from disk due to memory constraints.

**What is Over-Allocation and High Access Time?**Over-Allocation occurs when more memory is allocated than what is physically available in the system. This can lead to excessive swapping of pages between RAM and disk, resulting in degraded performance.  
  
High access time refers to the delay experienced when accessing data from memory or disk. It can be caused by factors such as high page faults, slow r/w speeds, or inefficient memory management algorithms.

**Mention the most common Page Replacement algorithms.**

* FIFO (First-In-First-Out): Replaces the oldest page in memory. Not good if the oldest page is frequently used.
* OPTIMAL: Replaces the page that will not be used for the longest period in the future. Though it looks good on paper, it is impractical in real life.
* LRU (Least Recently Used): Replaces the page that has not been used for the longest period. This is a practical algorithm.

## Chapter 9 – Secondary Memory Management

**What is Mass-Storage Structure?**Mass-Storage Structure refers to the organization and management of data storage devices such as HDD, SSD, and magnetic disks within a computer system.

**What are Magnetic Disks?**Magnetic disks are storage media that use magnetic particles to store data. They are commonly used for long-term storage of large amounts of data in computer systems.

**What is a Head Crash?**A head crash refers to a physical failure of a hard disk drive (HDD) where r/w heads contact the magnetic storage surface of the disk platters. This contact results in damage to the magnetic coating on the disk surface, potentially causing data loss.

**What is Constant Linear Velocity (CLV)?**Constant Linear Velocity is a method used in hard disks and also various optical storage devices, such as CDs and DVDs, where the speed of the disc’s rotation varies depending on the position of the r/w head.

**Explain how Disk Scheduling works.**Disk scheduling is a process of deciding the order of which disk I/O requests are serviced by a computer’s disk controller. It involves selecting the most efficient sequence for accessing data stored on the disk. This is done by using various Disk Scheduling algorithms.

**Mention the most common Disk Scheduling Algorithms**

* First-Come, First-Served (FCFS): Processes requests in the order they arrive.
* Shortest-Seek-Time-First (SSTF): Services the request with the shortest seek time from the current head position.
* SCAN (Elevator): Moves the disk arm back and forth across the disk, servicing requests in the direction of arm movement.
* Circular SCAN (C-SCAN): Same logic as SCAN, but disk head only moves one direction.
* LOOK: Like SCAN but reverses head direction before reaching the end if the remaining distance does not have any requests.
* Circular LOOK (C-LOOK): Same logic as LOOK, but only moves disk head one direction.

**What is Seek Time and Rotational Latency?**Seek time is the time it takes for the disk’s head to move to the track containing the requested data.  
Rotational Latency is the additional time it takes for the desired sector of the disk to rotate under the disk head once the head is positioned over the right track.

## Chapter 10 – File Systems

**What is a File System and why do we use it?**A File System is a method used by the OS to organize and store data on storage devices such as HDD, SSD, etc. It provides a logical structure for organizing files and directories, managing access to data, and controlling storage allocation and utilization. A file system defines how data is stored, named, accessed, and manipulated by users and apps.

**Mention some File Systems**

* FAT
* NTFS
* Ext
* HFS
* APFS

**What are the general file attributes?**

* Name
* Identifier
* Type
* Location
* Size
* Permission
* Creation date
* Last modified date
* Owner

**Why do we use Path names?**We use path names to specify the location of files and directories within a filesystem. This helps us navigate and access data stored on storage devices, enabling us to locate, open, modify, and manage files and directories.

**What are hard and soft links?**A Hard Link is a directory entry that points directly to the physical location of a file on the disk. Deleting the original file does not delete the hard link as it points directly to the file’s data.

A Soft Link is a file that contains a path to a file or directory. Unlike hard links, soft links point to the file’s path rather than its physical location. If the original file or directory is moved or deleted, the soft link will be broken.

**Mention some file sharing techniques.**

* FTP/SFPT
* NSF
* P2P
* USB drives
* Email
* Cloud storage

**What is the table through which Logical Addresses are translated to physical addresses called?**The table through which Logical Addresses are translated to physical addresses is called the Translation Lookaside Buffer (TLB). It is used to speed up virtual-to-physical address translation for accessing memory.

## Extra – Practical Questions

**Consider the virtual page reference string {4, 7, 6, 1, 7, 6, 1, 2, 7, 2}. In a demand-paged virtual memory system running on a computer with a main memory size of 3-page frames initially empty, determine the number of page faults under the corresponding page replacement algorithm policies: LRU, FIFO, and OPTIMAL.**

LRU: 6

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4 | 7 | 6 | 1 | 7 | 6 | 1 | 2 | 7 | 2 |
|  |  | 6 | 6 | 6 | 6 | 6 | 6 | 7 | 7 |
|  | 7 | 7 | 7 | 7 | 7 | 7 | 2 | 2 | 2 |
| 4 | 4 | 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| F | **F** | **F** | **F** |  |  |  | **F** | **F** |  |

FIFO: 6

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4 | 7 | 6 | 1 | 7 | 6 | 1 | 2 | 7 | 2 |
|  |  | 6 | 6 | 6 | 6 | 6 | 6 | 7 | 7 |
|  | 7 | 7 | 7 | 7 | 7 | 7 | 2 | 2 | 2 |
| 4 | 4 | 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| F | **F** | **F** | **F** |  |  |  | **F** | **F** |  |

OPT: 5

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4 | 7 | 6 | 1 | 7 | 6 | 1 | 2 | 7 | 2 |
|  |  | 6 | 6 | 6 | 6 | 6 | 2 | 2 | 2 |
|  | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| 4 | 4 | 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| F | **F** | **F** | **F** |  |  |  | **F** |  |  |

**Consider the code:**

public class Counter {

protected long count = 0;

public synchronized void add(long value) {

this.count = this.count + value;

}

}

**Two threads A and B both call add function and complete their execution. If thread A passes value=2 and Thread B passes value=3.**

**What is the value of “count” when both the threads finish?**

The count value will become “5” when both threads finish. Since the method is using the “synchronized” keyword, the order of when the threads are run does not matter.

**In the figure to the right, five processes and their burst time are given. Assume non-preemptive scheduling is followed and all the processes arrive at the same time T=0.**

A white rectangular box with black text

Description automatically generated**If process P2 wants to complete its execution with minimum waiting time, which scheduling technique will be best assuming none of the process needs IO devices and why?**

If process P2 wants to complete its execution with minimum waiting time, one scheduling technique will be Shortest Job First (SJF). This leaves P2 second in line with only P4 in front (which conveniently has the shortest burst time). Another scheduling technique is priority scheduling. Using this technique P2 can be assigned highest priority, and therefore not have to wait at all.

**b) After the scheduling approach is followed, calculate the waiting time for process P2?**

Since priority scheduling is as simple as just assigning a priority value, the waiting time for P2 will be 0. I contrast to SJF where its waiting time would have been 3.

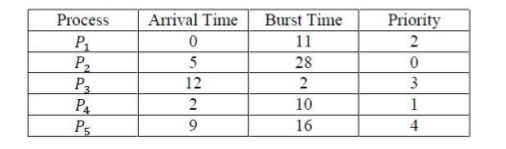
A diagram of a diagram of a network

Description automatically generated with medium confidence

**Consider the figure to the right. There are three processes T1, T2 and T3. There are four resources R1, R2, R3, R4. Resources R1 and R3 have one instance of resource each, while resource R2 has two instances and R4 has three instances of resource. Also, resources cannot be pre-empted by the processes. Will the situation lead to deadlock? If so, how?**

This situation will lead to a deadlock. The graph not only contains cycles, but also each resource category contains only a single instance.

**Consider five processes shown in the figure below:**



**Given the processes with Arrival time, CPU burst time and Priority (0 IS THE HIGHEST PRIORITY).**

**Calculate the average waiting time and average turnaround time if Preemtive Priority Scheduling is applied.**

**Average waiting time:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | P1 | P4 | P2 | P4 | P1 | P3 | P5 |  |
| Time | 0 | 2 | 5 | 33 | 40 | 49 | 51 | 66 |
| Gap | 0 | 2 | 3 | 28 | 7 | 9 | 2 | 15 |
| WT | 38 | 28 | 0 | - | - | 37 | 42 |  |

**Average turnaround time:**

**Considering a system with five processes P0 through P4 and three resources of type A, B, C. Resource type A has 10 instances, B has 5 instances and type C has 7 instances. Suppose at time t0 following snapshot of the system has been taken:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Process | Allocation | | | Max | | | Available | | |
| A | B | C | A | B | C | A | B | C |
| P0 | 0 | 1 | 0 | 7 | 5 | 3 | 3 | 3 | 2 |
| P1 | 2 | 0 | 0 | 3 | 2 | 2 |  |  |  |
| P2 | 3 | 0 | 2 | 9 | 0 | 2 |  |  |  |
| P3 | 2 | 1 | 1 | 2 | 2 | 2 |  |  |  |
| P4 | 0 | 0 | 2 | 4 | 3 | 3 |  |  |  |

**What will be the content of the Need matrix?**

|  |  |  |  |
| --- | --- | --- | --- |
| Process | Need | | |
| A | B | C |
| P0 | 7 | 4 | 3 |
| P1 | 1 | 2 | 2 |
| P2 | 6 | 0 | 0 |
| P3 | 0 | 1 | 1 |
| P4 | 4 | 3 | 1 |

Need = Max – Allocation

**Is the system in a safe state? If yes, then what is the safe sequence?**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Process | Allocation | | | Max | | | Available | | | Order |
| A | B | C | A | B | C | A | B | C |  |
| P0 | 0 | 1 | 0 | 7 | 5 | 3 | ~~3~~ | ~~3~~ | ~~2~~ | 4 |
| P1 | 2 | 0 | 0 | 3 | 2 | 2 | 5 | 3 | 2 | 1 |
| P2 | 3 | 0 | 2 | 9 | 0 | 2 |  |  |  | 5 |
| P3 | 2 | 1 | 1 | 2 | 2 | 2 | 7 | 4 | 3 | 2 |
| P4 | 0 | 0 | 2 | 4 | 3 | 3 | 7 | 4 | 5 | 3 |

Using the Safety algorithm, we find that the order is: P1 -> P3 -> P4 -> P0 -> P2

**Consider the queue with disk heads as 98, 183, 37, 122, 14, 124, 65, 67.**

**The Head starts at 53. The total disk head movement if SHORTEST SEEK TIME FIRST disk scheduling algorithm is followed is what?**

Order is chosen based on the closest distance to the current position. The difference is calculated to find the seek time.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Head | 98 | 183 | 37 | 122 | 14 | 124 | 65 | 67 |
| Order | 5 | 8 | 3 | 6 | 4 | 7 | 1 | 2 |

(65 - 53) + (67 - 65) + (67 - 37) + (37 - 14) + (98 - 14) + (122 - 98) + (124 - 122) + (183 - 124) = 236

Therefore, the total disk head movement using SSTF is 236.