ASSIGNMENT

OPERATING SYSTEM

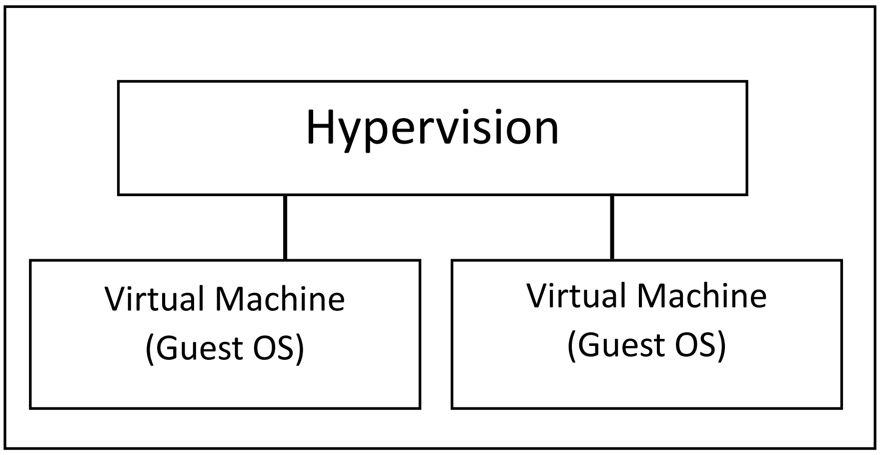
NAME: RAHUL ARYAL
E. ID:- ADTU/2021-25/BCS/015

Assignment

Operating System

1. **With a neat diagram explain the concept of virtual machine.**

   Ans: The execution of numerous operating systems and applications on a single physical machine is made possible by virtual machines (VMs), which are software emulations of real computers. The guest operating system and applications can operate independently of the underlying hardware in this isolated and self-contained environment.



2. **Distinguish between CPU bounded, I/O bounded processes.**

   Ans: The differences between CPU bounded and I/O bounded processes are:

   CPU bounded Process:

   i) Process spends most of its time using processor
   ii) Longer CPU Burst
   iii) Compiler, Simulator, Scientific application

   I/O Bounded Process:

   i) Process spends most of its time using I/O.
   ii) Shorter CPU burst
   iii) Word Processors, database applications.

3. **What is semaphore? Explain how to use them.**

   ✓ **Ans:** Semaphores are integers variables that are used to solve critical section problem by using two atomic operations, wait and signal that are used for process synchronization.

Rahul Aryal

Assignment

Operating System

To use Semaphore, following points are applied:

a. Initialize the Semaphore: Create and initialize a semaphore with an initial value. The initial value represents the number of concurrent threads or processes allowed to access the shared resource simultaneously.

b. Acquiring the Semaphore (Wait): A thread or process wanting to access the shared resource needs to acquire the semaphore. If the semaphore value is greater than zero, the thread can proceed, and the semaphore value is decremented. If the semaphore value is zero, indicating that all resources are currently being used, the thread may be blocked or put to sleep until the semaphore becomes available.

c. Releasing the Semaphore (Signal): After a thread or process has finished using the shared resource, it releases the semaphore by incrementing its value. This operation allows waiting threads or processes to proceed if any.

4. **How process switching, scheduling, and dispatching is implemented?**

Ans: Process switching, scheduling, and dispatching are key components of an operating system that manage the execution of multiple processes.

Process switching involves transferring the control of the CPU from one process to another. It typically occurs when a process needs to wait for an event, such as I/O completion or a timer interrupt. The operating system saves the current process's state and restores the state of the next process to be executed.

Scheduling determines the order in which processes are executed on the CPU. The scheduler selects processes from the ready queue based on scheduling algorithms such as First-Come, First-Served (FCFS), Round Robin, or Priority Scheduling. The goal is to optimize resource utilization, responsiveness, fairness, or a combination of these factors.

Dispatching refers to the actual act of transferring control to a selected process. The dispatcher performs necessary operations to prepare the CPU for executing the selected process, such as updating the process's PCB (Process Control Block), loading its program into memory, and setting up the execution environment. Afterward, the dispatcher transfers control to the chosen process, allowing it to execute.

Together, process switching, scheduling, and dispatching enable the operating system to efficiently manage the execution of processes, ensuring fair access to resources, responsiveness, and overall system stability.

Assignment

Operating System

5. **Explain the important aspects associated with deadlock avoidance.**

   Ans: The different important aspects associated with deadlock avoidance are:

   a) **Safe state:**
   A safe state is safe, if the system can allocate resources to each process in some order and avoid deadlock. Formally a system is said to be in safe state only if their exist a safe sequence. A sequence of processes <P1, P2, P3…….., Pn> is a safe sequence for the current allocation state if and only if for each process Pi , the resources that P1 can still request can be satisfied by the currently available resources plus the resources held by all the processes Pj with j>i.

   b) **Resource Allocation Graph:**
   A resource allocation graph is used to model the allocation and request relationships between processes and resources. By analysing the graph, potential deadlocks can be detected by identifying cycles.

   c) **Banker's algorithm:**
   The Banker's algorithm is an aspect associated with deadlock avoidance. The Banker's algorithm is a resource allocation and deadlock avoidance algorithm used in operating systems. It helps prevent the occurrence of deadlocks by ensuring that resource requests from processes will not result in an unsafe state.

   d) **Resource Request Protocols:**
   Processes must follow specific protocols when requesting resources to prevent deadlock. Protocols such as resource hierarchy, where processes must request resources in a predefined order, are implemented to avoid circular wait conditions.

6. **Why is paging faster than segmentation?**

   Ans: Paging is generally considered faster than segmentation due to the way memory is divided and accessed. In paging, memory is divided into fixed-size blocks called pages, while in segmentation, memory is divided into variable-sized logical segments.

   Paging offers several advantages that contribute to its speed. It allows for efficient use of physical memory by dividing it into equal-sized pages, simplifying memory management. Paging enables the use of a page table that maps virtual addresses to physical addresses, allowing for direct memory access. In contrast, segmentation requires additional translation mechanisms, such as segment tables, which introduce additional overhead.

   Additionally, paging enables the use of a concept called "page-level" protection and sharing, allowing for more efficient memory protection and inter-process communication. These factors combined make paging faster and more efficient for memory management compared to segmentation.

Assignment

Operating System

7. **What is super block explain?**

   Ans: A superblock is a collection of metadata used to show the properties of file systems in some types of operating systems. The superblock is one of a handful of tools used to describe a file system along with inode, entry and file. The superblock stores much of the information about the file system, which includes the following:

   - Size and status of the file system
   - Label (file system name and volume name)
   - Size of the file system logical block
   - Date and time of the last update
   - Cylinder group size
   - Number of data blocks in a cylinder group
   - Summary data block
   - File system state
   - Path name of the last mount point

8. **Differentiate between Micro kernel and Macro Kernel architecture?**

   Ans: The differences between Micro kernel and Macro Kernel Architecture are:

   Micro kernel:

   i) Kernel size is small and minimal.
   ii) Emphasis modularity and flexibility
   iii) Less complex due to minimal kernel
   iv) Improve fault isolation between components
   v) Eg: L4, QNX, Minix

   Macro kernel:

   i) Kernel size is larger and more comprehensive.
   ii) Less modularity and more tightly coupled.
   iii) More complex due to integrated services
   iv) Limited fault isolation
   v) Eg: Linux, Windows NT

9. **Is it better to create a child process rather than creating a thread for doing a function? Justify your answer.**

   Ans: The choice between creating a child process or a thread depends on the specific requirements and characteristics of the task at hand.

   Creating a child process offers advantages in terms of process isolation and fault tolerance. Each child process has its own memory space, file descriptors, and system resources, ensuring better protection and stability. Additionally, if one child process crashes, it does not affect the others.

On the other hand, threads are more lightweight and have lower overhead since they share the same memory space and resources of the parent process. Thread creation and synchronization are generally faster than process creation.
Ultimately, the decision depends on factors such as the level of isolation needed, resource sharing requirements, fault tolerance considerations, and performance trade-offs.

10. Describe the Bounded - buffer problem and give a solution for the same using semaphore

Ans: Bounded buffer means a finite pool of buffers in which each buffer holds one record of information. A producer process produces one record at a time and writes into the buffer. A consumer process consumes information one record at a time. A buffer is said to become full when a producer writes into it and is said to be empty when a consumer copies out a record contained in it.
Constraints applied on this problem for its solution are:

a) A producer must not overwrite a full buffer.
b) A consumer must not consume an empty buffer.
c) Producers and consumers must access buffers in a mutually exclusive manner.
d) Information must be consumed in the same order in which it is put into the buffers, i.e., FIFO order.

begin
Parbegin
    Var produced: boolean;
    repeat
 	
    produced = false;
    while produced = false
 	
if an empty buffer exists
then
 	
    {Produce in a buffer}
    produced = true
 	
{Remainder of the cycle}
forever ;
Parend ;
end;
	Var consumed: boolean;
	repeat
	consumed = false;
	while consumed = false
	
	if a buffer exists
	
	then
	
	    (consume a buffer}
	
	    consumed = true;
	
	{Remainder of the cycle}
	
	forever;

Producer	**Consumer**

Assignment

Operating System

11. **Consider the page reference string 1,3,4,0,5,3,2,1,0,4,5,2. How many page faults occur for the LRU and Optimal replacement algorithms with 4 frames each.**

Ans:

Ans: LRU

1	**3**	**4**	**0**	**5**	**3**	**2**	**1**	**0**	**4**	**5**	**2**
1	**1**	**1**	**1**	**5**	**5**	**5**	**5**	**0**	**0**	**0**	**0**
	3	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**4**	**4**	**4**
		4	**4**	**4**	**4**	**2**	**2**	**2**	**2**	**5**	**5**
			0	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**2**
Fault	**Fault**	**Fault**	**Fault**	**Fault**	**Hit**	**Fault**	**Fault**	**Fault**	**Fault**	**Fault**	**Fault**

Number of fault pages are: 11

Optimal replacement algorithm:

1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**4**	**4**	**4**
	3	**3**	**3**	**3**	**3**	**2**	**2**	**2**	**2**	**2**	**2**
		4	**4**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**
			0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Fault	**Fault**	**Fault**	**Fault**	**Fault**	**Hit**	**Fault**	**Hit**	**Hit**	**Fault**	**Hit**	**Hit**

Number of fault pages are: 7

12. **Why the page size is always of power of two? Differentiate between internal and external fragmentation?**

Ans: There is a problem in paging scheme that how does the compiler generate a 2d address. As we know that the compiler can generate only 1-d address in binary form. Then how is it possible to separate this address into two components 'p' and 'd?

For the solution of this problem, we take the size of pages in integral power of 2 such as 32, 64, ..., 1 K, 2K etc., because the single binary address can be shown to be the same as a 2-d array i.e., high order units correspond to 'p' and lower order units corresponds to 'd' which is stated earlier. This is the reason that compiler does not have to generate 2-d address for the paging system and it is the answer of the question that why are page sizes always powers of 2.

So, it generates only a single binary address, but it can be interpreted as a 2d address. This helps in separating 'p' from logical address and translating it to f and then concatenating the same 'd' to achieve the physical address.

Assignment

Operating System

Differences between internal and external fragmentation are:

Internal Fragmentation:

i) In internal fragmentation fixed-sized memory, blocks square measure appointed to process.

ii) Internal fragmentation happens when the method or process is smaller than the memory.

iii) The solution of internal fragmentation is the best-fit block.

iv) Internal fragmentation occurs when memory is divided into fixed-sized partitions.

External Fragmentation:

i) In external fragmentation, variable-sized memory blocks square measure appointed to the method.

ii) External fragmentation happens when the method or process is removed.

iii) The solution to external fragmentation is compaction and paging.

iv) External fragmentation occurs when memory is divided into variable size partitions based on the size of processes.

13. Consi[illegible] system with a set of processes P1, P2 and P3 and their CPU burst times, pri[illegible]es and arrival times being mentioned as below:

[illegible]rocess	CPU burst time	Arrival time	Priority
P1	5	0	2
P2	15	1	3
P3	10	2	1

Assuming 1 to be the highest priority and time quantum to be 2 units of time , calculate the average waiting time and turnaround time using FCFS , SJF, Priority (Pre-emptive and Non Pre-emptive), round robin scheduling mechanism .

Ans:

CT (Completion Time)

TAT (Turn Around Time) = CT – A T

WT (Waiting Time) = TAT – BT

a) Using FCFS Scheduling mechanism:

Process	CPU burst time	Arrival time	CT	TAT	WT
P1	5	0	5	5	0
P2	15	1	20	19	4
P3	10	2	30	28	18

Average waiting time => (0+4+18)/3 = 7.33
Average Turn Around Time => (5 + 19 + 28) = 17.33

b) SJF

Process	CPU burst time	Arrival time	CT	TAT	WT
P1	5	0	5	5	0
P2	15	1	30	29	14
P3	10	2	15	13	3

Average waiting time => (0+3+14)/3 = 5.67
Average Turn Around Time => (5 + 13 + 29)/3 = 15.67

c) Priority(Non-premptive)

Process	CPU burst time	Arrival time	CT	TAT	WT	Priority
P1	5	0	5	5	0	2
P2	15	1	20	19	4	3
P3	10	2	30	28	18	1

Average waiting time => (0+4+18)/3 = 7.33
Average Turn Around Time => (5 + 19 + 28)/3 = 17.33

d) Priority(Preemptive)

Process	CPU burst time	Arrival time	CT	TAT	WT	Priority
P1	5	0	15	15	10	2
P2	15	1	30	29	14	3
P3	10	2	12	10	0	1

Average waiting time => (10+14+0)/3 = 8
Average Turn Around Time => (15 + 29 + 10)/3 = 18

e) Round Robin:

Process	CPU burst time	Arrival time	CT	TAT	WT	Priority
P1	5	0	13	13	8	2
P2	15	1	30	29	14	3
P3	10	2	25	23	13	1

Average waiting time => (8+14+13)/3 = 11.67
Average Turn Around Time => (13 + 29 + 23)/3 = 21.67

Assignment

Operating System

14. Explain OS as a resource manager?

Ans: The operating system provides for an orderly and controlled allocation of the processors, memories, and I/O devices among the various programs in the bottom-up view. Operating system allows multiple programs to be in memory and run at the same time. Resource management includes multiplexing or sharing resources in two different ways: in time and in space. In time multiplexed, different programs take a chance of using CPU. First one tries to use the resource, then the next one that is ready in the queue and so on. For example: Sharing the printer one after another.The operating system provides for an orderly and controlled allocation of the processors, memories, and I/O devices among the various programs in the bottom-up view. Operating system allows multiple programs to be in memory and run at the same time. Resource management includes multiplexing or sharing resources in two different ways: in time and in space. In time multiplexed, different programs take a chance of using CPU. First one tries to use the resource, then the next one that is ready in the queue and so on. For example: Sharing the printer one after another.

✓

15. Distinguish between CPU bounded, I/O bounded processes.

Ans: Ans: The differences between CPU bounded and I/O bounded processes are:

CPU bounded Process:

i) Process spends most of its time using processor
ii) Longer CPU Burst
iii) Compiler, Simulator, Scientific application

I/O Bounded Process:

i) Process spends most of its time using I/O.
ii) Shorter CPU burst
iii) Word Processors, database applications

✓

16. What is the problem of busy wait?

Ans: The problem of busy wait, or spinning, occurs when a program or thread repeatedly checks for a condition without yielding the CPU. This leads to high CPU utilization and inefficient resource usage. The constant looping wastes computational resources and increases power consumption. It can also cause resource contention, as multiple threads may compete for the same resource, leading to potential deadlocks or starvation. Additionally, busy waiting prevents other tasks from executing, limiting the overall system performance. To mitigate this problem, alternative synchronization mechanisms, such as sleep/wait or event-driven notifications, should be employed to free up CPU resources and allow other processes to run efficiently.

Assignment

Operating System

17. Illustrate how demand paging affects system performance

Ans: Demand paging has several effects on system performance. By selectively loading only the necessary pages into memory, demand paging conserves memory resources and promotes efficient utilization. This capability enables the system to handle larger programs and multiple processes concurrently, ultimately enhancing overall system scalability.

Nonetheless, demand paging introduces an additional burden in the form of page faults. Whenever a page is accessed but not present in memory, a page fault occurs, necessitating a disk I/O operation to retrieve the required page. This disk I/O operation introduces latency and can have a substantial impact on system performance, especially if page faults occur frequently.

To mitigate the performance implications of demand paging, operating systems implement various strategies. These approaches involve optimizing page replacement algorithms to minimize page faults, employing techniques like pre-fetching and read-ahead to anticipate future page accesses, and utilizing efficient disk caching mechanisms to reduce the overhead of disk I/O operations.

Overall, demand paging facilitates efficient memory management, albeit with the trade-off of page faults. By carefully balancing memory usage, disk I/O, and page replacement strategies, system performance can be optimized, ensuring that the advantages of demand paging outweigh any potential drawbacks.

18. E [illegible] difference between pre-emptive and non-preemptive scheduling?

Ans: The differences between pre-emptive and non-preemptive scheduling are:

Pre-emptive:

i) In this resources(CPU Cycle) are allocated to a process for a limited time.
ii) Process can be interrupted in between.
iii) It has overheads of scheduling the processes.
iv) In preemptive scheduling, CPU utilization is high.
v) Preemptive scheduling waiting time is less.

Non Pre-emptive:

i) Once resources(CPU Cycle) are allocated to a process, the process holds it till it completes its burst time or switches to waiting state.
ii) Process can not be interrupted until it terminates itself or its time is up.
iii) It does not have overheads.
iv) It is low in non preemptive scheduling.
v) Non-preemptive scheduling waiting time is high.

Assignment

Operating System

19. How does an interrupt differ from a trap?

Ans: The trap is a signal raised by a user program instructing the operating system to perform some functionality immediately. In contrast, the interrupt is a signal to the CPU emitted by hardware that indicates an event that requires immediate attention. A trap also triggers OS functionality. It gives control to the trap handler. In contrast, an interrupt triggers the CPU to perform the interrupt handler routine. A trap also triggers OS functionality. It gives control to the trap handler. In contrast, an interrupt triggers the CPU to perform the interrupt handler routine. A trap is synchronous and may occur after the execution of the instruction. In contrast, an interrupt is asynchronous and may occur at any time. A trap is generated by a user program instruction. In contrast, the hardware devices generate an interrupt. A trap is also known as a software interrupt. In contrast, an interrupt is known as a hardware interrupt. The trap is a synchronous process. In contrast, the interrupt is an asynchronous process.

✓ **20. What are the use of job queues, ready queues and device queues**

Ans:

The Job Queue stores all processes that are entered into the system.
The Ready Queue holds processes in the ready state.
Device Queues hold processes that are waiting for any device to become available. For each I/O device, there are separate device queues.

21. Define Seek Time and Latency Time

Ans:

Seek Time:

Seek time refers to the time taken for the read/write heads of a hard disk drive (HDD) to move to the desired track or cylinder where data is stored. It represents the mechanical delay in accessing data and is measured in milliseconds (ms). A lower seek time indicates faster access to data, resulting in improved HDD performance.

Latency Time:

Latency time, also known as rotational latency or rotational delay, is the time it takes for the desired data sector on a hard disk to rotate under the read/write heads after a seek operation. It is determined by the rotational speed of the disk, measured in revolutions per minute (RPM). Lower latency time means faster data access, which can greatly impact overall disk performance, especially for random access operations.

Assignment

Operating System

22. Discuss in detail about the evolution of the Operating System?

Ans:

Serial Processing: The earliest computers executed tasks sequentially, without any OS. Users interacted directly with the machine using low-level instructions.

Batch Processing: With the introduction of batch systems, users submitted jobs in batches, and the OS scheduled and executed them automatically. Operators managed the system through consoles or punch cards.

Multiprogramming: Multiprogramming allowed concurrent execution of multiple programs, improving resource utilization. OSs introduced memory management techniques like overlays and swapping to handle limited memory.

Multi-tasking / Timesharing: Timesharing systems enabled multiple users to interact with a computer simultaneously. Users were given time slices, and the OS managed task switching and resource allocation.

Real-Time Operating System: Real-time operating systems (RTOS) are used in environments where a large number of events, mostly external to the computer system, must be accepted and processed in a short time or within certain deadlines. such applications are industrial control, telephone switching equipment, flight control, and real-time simulations. With an RTOS, the processing time is measured in tenths of seconds.

Distributed Operating System: A distributed operating system is system software over a collection of independent software, networked, communicating, and physically separate computational nodes. They handle jobs which are serviced by multiple CPUs. Each individual node holds a specific software subset of the global aggregate operating system.

23. What is the relation between effective access time and PFR?

Ans: The relation between Effective Access Time (EAT) and Page Fault Rate (PFR) in a virtual memory system can be summarized as follows:

EAT = (1 - PFR) * Memory Access Time + PFR * Page Fault Time

In this equation, EAT represents the average time taken to access a memory location, taking into account both memory hits and page faults. PFR represents the frequency of page faults, indicating the rate at which data needs to be fetched from secondary storage. The relation shows that as the page fault rate increases, the effective access time also increases. This is because page faults involve slower disk I/O operations, contributing to longer access times and impacting overall system performance. Reducing the page fault rate is important for minimizing the effective access time and improving system efficiency.

Assignment

Operating System

24. State the critical section problem with the help of an example?

Ans: A critical section for a data item ds is a section of code that shouldn't be executed concurrently either with itself or with other critical sections for the shared data ds.

Suppose P1 is a process and the Critical Section is assigned to the P1. Now, if P2 is requesting to enter into the critical section to perform some task, then P2 needs to be a wait because P1 is already using the critical section.

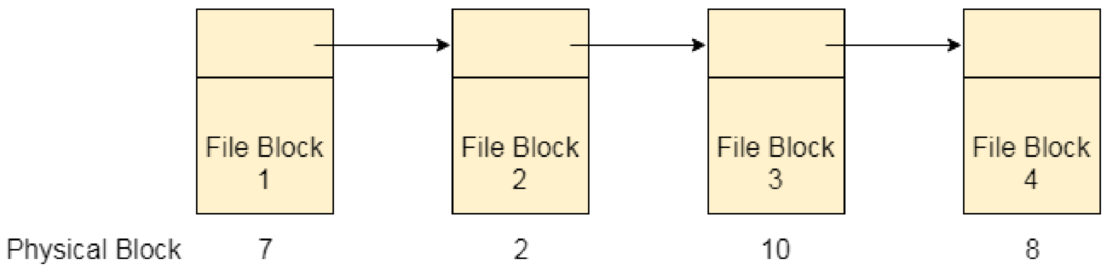
25. Discuss the concept of dynamic loading and linking with respect to memory management.

Ans: A linker links modules together to form a executable program. A static linker links all modules of a program before its execution begins whereas a dynamic linker links a module only when it is referenced.

A loader assigns memory locations to a program or part of a program. Dynamic loader loads only the referenced modules of the program. If a module is not needed it is not given the memory locations.
There by dynamic linking and loading makes efficient use of memory.

26. Briefly explain linked list file allocation method

Ans: Linked List allocation solves all problems of contiguous allocation. In linked list allocation, each file is considered as the linked list of disk blocks. However, the disks blocks allocated to a particular file need not to be contiguous on the disk. Each disk block allocated to a file contains a pointer which points to the next disk block allocated to the same file.



Linked List Allocation

27. Discuss the principle of direct memory access

Ans: The principle of direct memory access (DMA) is a technique used in computer systems to transfer data directly between peripheral devices and memory without involving the CPU. It enhances the efficiency of data transfers by offloading the CPU from the task of managing data movement.

In DMA, a DMA controller takes control of the bus and coordinates the transfer between the peripheral device and memory. The DMA controller accesses the data from the peripheral device, stores it in a buffer, and transfers it directly to the designated memory location. Similarly, it can also retrieve data from memory and transfer it to the peripheral device.

By bypassing the CPU, DMA significantly improves data transfer speeds and reduces CPU overhead. It is commonly used in scenarios where high-speed data transfers are required, such as disk I/O, network communication, and audio/video processing. DMA reduces latency and allows the CPU to focus on other tasks, enhancing system performance and responsiveness.

28. Describe the implementation of interposes communication using shared memory and message passing approaches?

✓

Ans: Interprocess communication (IPC) allows processes to communicate and exchange data in an operating system. Two common approaches for implementing IPC are shared memory and message passing.

In shared memory communication, a region of memory is shared among multiple processes. The implementation involves creating a shared memory segment that can be accessed by the participating processes. Processes attach themselves to the shared memory segment and coordinate access using synchronization mechanisms like semaphores or mutexes. This allows processes to read from and write to the shared memory, facilitating communication.

On the other hand, message passing communication involves processes sending and receiving messages. Processes establish communication channels such as pipes, sockets, or message queues. They can send messages containing data, requests, or signals through these channels. Synchronization mechanisms like locks or semaphores are used to coordinate message exchanges and ensure proper sequencing.

Shared memory provides a high-performance, low-overhead communication method as processes can directly access the shared memory. However, it requires careful synchronization to prevent data corruption. Message passing, on the other hand, offers a more structured and controlled form of communication, simplifying synchronization. It allows for better encapsulation of data and is suitable for scenarios where processes need to exchange discrete units of information.
The choice between shared memory and message passing depends on factors such as the nature of communication, performance requirements, and ease of implementation in a specific system. Both approaches have their advantages and can be utilized based on the specific IPC needs of an application.

Assignment

Operating System

29. Explain Banker's deadlock-avoidance algorithm with an illustration

Ans: Banker's Algorithm is a banker algorithm used to avoid deadlock and allocate resources safely to each process in the computer system.
To understand the Banker's Algorithm first we will see a real word example of it.

Suppose the number of account holders in a particular bank is 'n', and the total money in a bank is 'T'. If an account holder applies for a loan; first, the bank subtracts the loan amount from full cash and then estimates the cash difference is greater than T to approve the loan amount. These steps are taken because if another person applies for a loan or withdraws some amount from the bank, it helps the bank manage and operate all things without any restriction in the functionality of the banking system.

30. Differentiate between paging and segmentation?

Ans:
Difference between Paging and Segmentation are:

Paging:

i) In paging, the program is divided into fixed or mounted size pages.
ii) For the paging operating system is accountable.
iii) Page size is determined by hardware.
iv) It is faster in comparison to segmentation.
v) Paging could result in internal fragmentation.

Segmentation:

i) In segmentation, the program is divided into variable size sections.
ii) For segmentation compiler is accountable.
iii) Here, the section size is given by the user.
iv) Segmentation is slow.
v) Segmentation could result in external fragmentation.

31. Consider a logical address space of eight pages of 1024 words, each mapped onto a physical memory of 32 frames then calculate how many bits are in the logical address and physical address?

Ans:
Given,
Number of pages = $8 = 2^3$ and number of frames = $32 = 2^5$
Size of each frame = size of each page = $1024 = 2^{10}$
We know that,
Number of frames = size of Physical memory / size of each frame
So, size of Physical memory = $2^{10} * 2^5 = 2^{15}$ = 15 bits.
Number of processes = size of Logical memory / size of each process
So, size of Logical memory = $2^{10} * 2^3 = 2^{13}$ = 13 bits.

Assignment

Operating System

32. Discuss the criteria for choosing a file organization

Ans: When choosing a file organization in an operating system, several criteria should be considered to optimize storage and access efficiency. The following are some key criteria for selecting a file organization:

1. Performance: One of the primary factors to consider is the performance of file access operations. This includes the speed of file reads and writes, as well as the efficiency of searching and updating files. Different file organizations have varying impacts on performance, so the chosen organization should align with the expected usage patterns and performance requirements.

2. Space utilization: Efficient utilization of storage space is crucial. Some file organizations may result in wasted space due to internal fragmentation or excessive metadata overhead. It is important to select a file organization that minimizes wasted space and maximizes overall storage efficiency.

3. Scalability: The chosen file organization should be scalable to accommodate the expected growth in the number and size of files. It should handle a large number of files efficiently and support expansion without significant performance degradation.

4. File size and type: The nature of the files being stored can influence the choice of file organization. For example, if the files are predominantly large, sequential access-oriented files, a different organization may be more suitable compared to a scenario with many small, randomly accessed files.

5. File access patterns: Understanding the access patterns of the files is crucial. If files are frequently accessed sequentially, a sequential file organization may be optimal. On the other hand, if there is a mix of sequential and random accesses, a different organization, such as an indexed file organization, might be more suitable.

33. Define Belady's Anomaly?

Ans: Belady's Anomaly is a phenomenon that occurs in some page replacement algorithms. In this anomaly, increasing the number of frames allocated to a process can cause an increase in the number of page faults. This contradicts the common intuition that more memory should reduce the number of page faults.

Assignment

Operating System

34. Consider the following snapshot of a system:

	Allocation				**Max**				**Available**			
Process	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**
Po	**0**	**0**	**1**	**2**	**0**	**0**	**1**	**2**	**1**	**5**	**2**	**0**
P1	**1**	**0**	**0**	**0**	**1**	**7**	**5**	**0**				
P2	**1**	**3**	**5**	**4**	**2**	**3**	**5**	**6**				
P3	**0**	**6**	**3**	**2**	**0**	**6**	**5**	**2**				
P4	**0**	**0**	**1**	**4**	**0**	**6**	**5**	**6**				

Is the system in safe state?

Ans:
We know that,
Need[i, j] = Max[i, j] – Allocation[i, j]. So, Need:

Process	**A**	**B**	**C**	**D**
P0	**0**	**0**	**0**	**0**
P1	**0**	**7**	**5**	**0**
P2	**1**	**0**	**0**	**2**
P3	**0**	**0**	**2**	**0**
P4	**0**	**6**	**4**	**2**

Also, for Pi if need <= available, then pi is in safe sequence
Available = available + allocation

For P0, Need = 0 0 0 0, Available = 1 5 2 0
Here, condition is True. So, Available = 1 5 2 0 + 0 0 1 2 = 1 5 3 2

For P1, Need = 0 7 5 0, Available = 1 5 3 2
Here, Condition is false. P1 must wait.

For P2, Need = 1 0 0 2, Available = 1 5 3 2
Here, condition is true. So, Available = 1 5 3 2 + 1 3 5 4 = 2 8 8 6

For P3, Need = 0 0 2 0, Available = 2 8 8 6
Here, condition is true. So, Available = 2 8 8 6 + 0 6 3 2 = 2 14 11 8

For P4, Need = 0 6 4 2, Available = 2 14 11 8
Here, condition is true. So, Available = 2 14 11 8 + 0 0 1 4 = 2 14 12 12

For P1, Need = 0 7 5 0, Available = 2 14 12 12
Here, condition is true. So, Available = 2 21 17 12

So, the process is in Safe State.

Assignment

Operating System

35. Outline a solution using semaphores to solve dining philosopher problem?

Ans:

```
repeat
   successful = false;
   while (not successful)
             if both forks are available then left the forks one at a time
             successful = true;
   if successful = false
   then
             block (P);
{eat);
             Put down both forks;
if left neighbour is waiting for his right fork
then
             activate (left neighbour);
if right neighbour is waiting for his left fork then
             activate (right neighbour);
   {think);
   Forever;
```

36. With the help of a neat diagram explain the various steps of address binding?

Ans: Address binding involves the process of associating a symbolic name (such as a variable or function) with a specific memory address. The steps are as follows:

Compilation: The source code is compiled into object code, which contains placeholders for memory addresses.

a) Linking: Object code is linked with external libraries to generate an executable file. Symbolic references are resolved to their memory addresses.

b) Loading: The operating system loads the executable into memory. The loader performs dynamic address binding, assigning actual memory addresses to the placeholders.

c) Execution: The program starts running, and memory addresses are accessed based on the bindings established during loading.

37. Consider the following page replacement string

7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

38. Assuming there are three memory frames, how many page faults would occur in case of LRU and Optimal page replacement algorithm

Ans:

12 9

Rahul Aryal

Assignment

Operating System

i) LRU

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2	2	4	4	4	0	0	0	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0
		1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7
F	F	F	F	H	F	~~F~~ H	F	F	F	F	H	H	F	H	F	H	F	H	H

Total Number of Page Faults are: ~~13~~ 12

ii) Optimal

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	2	7	7	7
	0	0	0	0	0	0	4	4	4	0	0	0	0	0	0	0	0	0	0
		1	1	1	3	3	3	3	3	3	3	3	1	1	1	1	1	1	1
F	F	F	F	H	F	H	F	H	H	F	H	H	F	H	H	H	F	H	H

Total number of page fault = 9

✓ **39. Differentiate between system software and application software?**

Ans: Differences between system software and application software are:

System software:

i) System Software maintains the system resources and gives the path for application software to run.
ii) Low-level languages are used to write the system software.
iii) It is general-purpose software.
iv) Without system software, the system stops and can't run.
v) Example: System software is an operating system, etc.

Application software:

i) Application software is built for specific tasks.
ii) While high-level languages are used to write the application software.
iii) While it is a specific purpose software.
iv) While Without application software system always runs.
v) Example: Application software is Photoshop, VLC player, etc.

Assignment

Operating System

✓ **40. Explain Process Control Block. Draw the block diagram of process transition states**

Ans:

The Process Control Block (PCB), also known as the Task Control Block (TCB), is a data structure used by operating systems to store and manage information about a running process. The PCB contains essential details related to a process, allowing the operating system to effectively manage and control its execution.

The PCB typically includes the following information:

a. Process Identifier (PID): A unique identifier assigned to each process.
b. Program Counter (PC): The address of the next instruction to be executed.
c. CPU Registers: Stores the values of CPU registers, such as accumulator, stack , and index registers.
d. Process State: Indicates the current state of the process (e.g., running, ready, blocked).
e. Priority: The priority level assigned to the process for scheduling purposes.
f. Memory Management Information: Information about the memory allocated to the process, such as base and limit registers.
g. I/O Status Information: Details about I/O devices used by the process, including open files and pending I/O requests.
h. Accounting Information: Information related to resource usage, execution time, and other statistics.

Now, let's consider the block diagram of process transition states:

New
Ready
Running
Blocked
Terminated

a. New: The process is being created or initialized.
b. Ready: The process is in main memory and waiting to be assigned to a processor.
c. Running: The process is currently being executed by a processor.
d. Blocked: The process is waiting for a particular event (such as I/O completion) before it can proceed.
e. Terminated: The process has completed its execution or been explicitly terminated.

These transition states illustrate the life cycle of a process, showing how a process can move between different states based on the actions it performs or the events it encounters. The operating system utilizes the PCB to manage these state transitions and perform various process scheduling and resource allocation decisions based on the information stored within the PCB.

41. What is the role of critical section in process management?
Ans:
The role of the critical section can be summarized as follows:

1. Mutual Exclusion: The primary role of the critical section is to provide mutual exclusion, ensuring that only one process at a time can execute the critical section code. This prevents concurrent access to shared resources, avoiding conflicts and maintaining data integrity.

2. Synchronization: The critical section facilitates synchronization among processes by enforcing a specific order of execution. It ensures that processes wait their turn to access shared resources, preventing race conditions and maintaining the desired behavior of the program.

3. Data Consistency: Shared resources, such as variables, data structures, or I/O devices, can be accessed and modified by multiple processes. The critical section ensures that only one process can modify the shared resource at a time, preventing inconsistent or incorrect data states.

4. Atomicity: The critical section ensures that the execution of a set of operations on shared resources appears as an indivisible or atomic action. This means that the operations are treated as a single, uninterrupted unit, preventing intermediate states that could lead to incorrect behavior or corruption of data.

42. What is RAG? Explain how it is very useful in describing deadly embrace by considering an example?
Ans: RAG stands for Resource Allocation Graph. It is a graphical representation used in operating systems to illustrate the allocation and availability of resources and the relationships between processes and resources. The RAG is particularly useful in analyzing deadlock situations and resource allocation problems.

Consider an example with three processes (P1, P2, P3) and three resources (R1, R2, R3). The following allocation and request relationships exist:

P1 holds R2 and is requesting R3.
P2 holds R3 and is requesting R1.

P3 holds R1 and is requesting R2.

In the RAG, each process is represented by a node (P1, P2, P3), and each resource is represented by a node (R1, R2, R3). The allocation and request relationships are depicted by directed edges.

Looking at the RAG, we can observe that there is a circular dependency or cycle formed among P1, P2, and P3. Each process is waiting for a resource held by another process in the cycle. This circular wait leads to a deadly embrace, causing a deadlock situation where none of the processes can progress.

By analyzing the RAG and recognizing such circular dependencies, we can identify the presence of a deadly embrace and take appropriate measures to resolve the deadlock. Strategies for resolving the deadlock may include resource preemption, where a resource is forcibly taken from one process and allocated to another, or employing techniques like resource ordering or avoiding circular wait scenarios during system design.

In summary, the Resource Allocation Graph is immensely useful in describing a deadly embrace by visually representing the circular dependency among processes and resources. It helps in understanding the deadlock scenario and facilitates the development of effective deadlock resolution strategies.

43. Illustrate with example, the internal and external fragmentation problem?
Ans: Certainly! Let's illustrate the concepts of internal and external fragmentation with an example scenario.

Internal Fragmentation:
Suppose we have a disk system with a block size of 1 KB. We need to store three files: File A (1.5 KB), File B (2 KB), and File C (800 bytes). Each file is allocated to a separate block. In this case, there is internal fragmentation because each file is allocated a whole block, resulting in unused space within the blocks. For example, in File A's block, 500 bytes remain unused, which is wasted space.

External Fragmentation:
Consider a memory system with variable-sized memory blocks. Initially, the memory has three blocks: Block 1 (4 KB), Block 2 (2 KB), and Block 3 (6 KB). Over time, processes are allocated and deallocated, resulting in blocks of varying sizes scattered throughout the memory. Eventually, the memory becomes fragmented, with free blocks interspersed but not contiguous. This leads to external fragmentation, as even though sufficient total free memory may exist, it cannot be used to allocate larger processes due to the lack of contiguous memory space.

Both internal and external fragmentation hinder efficient utilization of resources and can be addressed through techniques such as compaction, dynamic memory allocation, or file system optimization.

44. Consider the following reference string : 1,2,3,,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6
How many pages faults will occur for (i) FIFO and (ii) LRU page replacement algorithms with 3 frames.
Ans:
FIFO

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	1	4	4	4	4	6	6	6	6	3	3	3	3	3	1	1	1	1
	2	2	2	2	1	1	1	2	2	2	2	7	7	7	7	7	7	3	3
		3	3	3	3	5	5	5	1	1	1	1	6	6	2	2	2	2	6
F	F	F	F	H	F	F	F	F	F	H	F	F	F	H	F	F	H	F	F

Total page faults are: 16

LRU

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	1	4	4	4	5	5	5	1	1	1	7	7	7	2	2	2	2	2
	2	2	2	2	2	2	6	6	6	6	3	3	3	3	3	3	3	3	3
		3	3	3	1	1	1	2	2	2	2	2	6	6	6	1	1	1	6
F	F	F	F	H	F	F	F	F	F	H	F	F	F	H	F	F	H	H	F

Total page faults are: 15

45. Consider the track requests in the disk queue (23, 89, 132, 42, 187), head starts at position 100. Explain and compute the total head movement using the following disk scheduling algorithms. (i) FCFS (ii) SCAN

Ans: (i) FCFS (First-Come, First-Served):
Given the track requests (23, 89, 132, 42, 187) and the head starting at position 100, the total head movement can be calculated as follows:

Head Movement = |Current Position - Track Request 1| + |Track Request 1 - Track Request 2| + |Track Request 2 - Track Request 3| + |Track Request 3 - Track Request 4| + |Track Request 4 - Track Request 5|

Head Movement = |100 - 23| + |23 - 89| + |89 - 132| + |132 - 42| + |42 - 187|
Head Movement = 77 + 66 + 43 + 90 + 145
Head Movement = 421
Therefore, the correct total head movement using FCFS disk scheduling is 421.

(ii) SCAN:
Given the track requests (23, 89, 132, 42, 187) and the head starting at position 100, the total head movement can be calculated as follows:
Move the head towards the higher-numbered tracks until the highest requested track (in this case, track 187) is reached.
Reverse the direction and move the head towards the lower-numbered tracks until the lowest requested track (in this case, track 23) is reached.
Head Movement = (Track 187 - Track 100) + (Track 187 - Track 23)
Head Movement = (187 - 100) + (187 - 23)
Head Movement = 87 + 164
Head Movement = 251