OPERATING SYSTEMS

Unit1_Unit2_Unit3:Revision Class #5 Typical Questions related to Operating Systems

Nitin V Pujari
Faculty, Computer Science
Dean , IQAC, PES University

OPERATING SYSTEMS

Course Syllabus => Unit 1

1	Introduction: What Operating Systems Do, Computer-System Organization	1.1, 1.2	21.4
2	Computer-System Architecture, Operating-System Structure & Operations	1.3,1.4,1.5	
3	Kernel Data Structures, Computing Environments	1.10, 1.11	
4	Operating-System Services, Operating-System Design and Implementation	2.1, 2.6	
5	Process concept: Process in memory, Process State, Process Control Block, Context switch, Process Creation & Termination,	3.1 - 3.3	
6	CPU Scheduling - Preemptive and Non-Preemptive, Scheduling Criteria, FIFO Algrorithm	5.1-5.2	
7	Scheduling Algorithms:SJF, Round-Robin and Priority Scheduling	5.3	
8	Multi-Level Queue, Multi-Level Feedback Queue	5.3	
9	Multiprocessor and Real Time Scheduling	5.5, 5.6	
10	Case Study: Linux/ Windows Scheduling Policies.	5.7	
11	Inter Process Communication - Shared Memory, Messages	3.4	
12.	Named and unnamed pipes (+Review)	3.6.3	

OPERATING SYSTEMS

Course Syllabus => Unit 2

13	Introduction to Threads, types of threads, Multicore Programming, Multithreading Models	4.1 - 4.3	42.8
14	Thread creation, Thread Scheduling	5.4	
15	Pthreads and Windows Threads	4.4	
16	Mutual Exclusion and Synchronization: software approaches,	6.1-6.2	
17	principles of concurrency, hardware support	6.3-6.4	
18	Mutex Locks, Semaphores	6.5, 6.6	
19	Classic problems of Synchronization: Bounded-Buffer Problem, Readers-Writers problem	6.7-6.8	
20	Dining-Philosophers Problem	6.8	
21	Synchronization Examples: Synchronisation mechanisms provided by Linux/Windows/Pthreads.	6.9	
22	Deadlocks: principles of deadlock, Deadlock Characterization	7.1-7.3	
23	Deadlock Prevention, Deadlock example	7.4-7.5	
24	Deadlock Detection, Algorithm	7.6	

OPERATING SYSTEMS

Course Syllabus => Unit 3

25	Main Memory: Hardware and control structures, OS support, Address translation	8.1	64.2
26	Dynamic linking, Swapping	8.2	
27	Memory Allocation (Partitioning, relocation), Fragmentation	8.3	
28	Segmentation	8.4	
29	Paging: OS Support, TLBs, Address Translation	8.5	
30	Structure of the Page Table	8.6	
31	Design Alternatives - Inverted Page Tables, Bigger Pages	8.7-8.8	
32	Virtual Memory: Demand Paging, Copy-OnWrite	9.1-9.3	
33	Page replacement policy - LRU etc. (In comparison with FIFO and Optimal)	9.4	
34	Page Replacement (contd.), Frame allocation	9.4,9.5	
35	Thrashing	9.6	
36	Case Study: Linux/ Windows Memory Management	9.10	

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

What are the three main purposes of an operating system?

We have stressed the need for an operating system to make efficient use of the computing hardware. When is it appropriate for the operating system to forsake this principle and to "waste" resources? Why is such a system not really wasteful?

What is the main difficulty that a programmer must overcome in writing an operating system for a real-time environment?

Keeping in mind the various definitions of **operating system,** consider whether the operating system should include applications such as Web browsers and mail programs. Argue both that it should and that it should not, and support your answers.

How does the distinction between kernel mode and user mode function as a rudimentary form of protection (security) system?

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

Which of the following instructions should be privileged?

a. Set value of timer.

b. Read the clock.

c. Clear memory.

d. Issue a trap instruction.

e. Turn off interrupts.

f. Modify entries in device-status table.

g. Switch from user to kernel mode.

h. Access I/O device.

Some early computers protected the operating system by placing it in a memory partition that could not be modified by either the user job or the operating system itself. Describe two difficulties that you think could arise with such a scheme.

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

Some CPUs provide for more than two modes of operation. What are two possible uses of these multiple modes?

Timers could be used to compute the current time. Provide a short description of how this could be accomplished.

Give two reasons why caches are useful. What problems do they solve? What problems do they cause? If a cache can be made as large as the device for which it is caching (for instance, a cache as large as a disk), why not make it that large and eliminate the device?

Distinguish between the client–server and peer-to-peer models of distributed systems.

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

What is the purpose of system calls?

What are the five major activities of an operating system with regard to process management?

What are the three major activities of an operating system with regard to memory management?

What are the three major activities of an operating system with regard to secondary-storage management?

What is the purpose of the command interpreter? Why is it usually separate from the kernel?

What system calls have to be executed by a command interpreter or shell in order to start a new process?

What is the purpose of system programs?

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

What is the main advantage of the layered approach to system design? What are the disadvantages of using the layered approach?

List five services provided by an operating system, and explain how each creates convenience for users. In which cases would it be impossible for user-level programs to provide these services? Explain your answer.

Why do some systems store the operating system in firmware, while others store it on disk?

How could a system be designed to allow a choice of operating systems from which to boot? What would the bootstrap program need to do?

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

When a process creates a new process using the fork() operation, which of the following state is shared between the parent process and the child process?

a. Stack
b. Heap
c. Shared memory segments

Provide three programming examples in which multithreading provides better performance than a single-threaded solution.

What are two differences between user-level threads and kernel-level threads? Under what circumstances is one type better than the other?

Describe the actions taken by a kernel to context-switch between kernel-level threads.

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

What resources are used when a thread is created? How do they differ from those used when a process is created?

Assume that an operating system maps user-level threads to the kernel using the many-to-many model and that the mapping is done through LWPs. Furthermore, the system allows developers to create real-time threads for use in real-time systems. Is it necessary to bind a real-time thread to an LWP? Explain.

we mentioned that disabling interrupts frequently can affect the system's clock. Explain why this can occur and how such effects can be minimized.

Explain why Windows, Linux, and Solaris implement multiple locking mechanisms. Describe the circumstances under which they use spin-locks, mutex locks, semaphores, adaptive mutex locks, and condition variables. In each case, explain why the mechanism is needed.

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

What is the meaning of the term **busy waiting**? What other kinds of waiting are there in an operating system? Can busy waiting be avoided altogether? Explain your answer.

Explain why spinlocks are not appropriate for single-processor systems yet are often used in multiprocessor systems.

Show that, if the `wait()` and `signal()` semaphore operations are not executed atomically, then mutual exclusion may be violated.

Illustrate how a binary semaphore can be used to implement mutual exclusion among *n* processes.

Can you answer the following typical questions related to Operating Systems ?

A CPU-scheduling algorithm determines an order for the execution of its scheduled processes. Given n processes to be scheduled on one processor, how many different schedules are possible? Give a formula in terms of n.

Explain the difference between preemptive and nonpreemptive scheduling.

What advantage is there in having different time-quantum sizes at different levels of a multilevel queueing system?

Many CPU-scheduling algorithms are parameterized. For example, the RR algorithm requires a parameter to indicate the time slice. Multilevel feedback queues require parameters to define the number of queues, the scheduling algorithms for each queue, the criteria used to move processes between queues, and so on.

These algorithms are thus really sets of algorithms (for example, the set of RR algorithms for all time slices, and so on). One set of algorithms may include another (for example, the FCFS algorithm is the RR algorithm with an infinite time quantum). What (if any) relation holds between the following pairs of algorithm sets?

a. Priority and SJF
b. Multilevel feedback queues and FCFS
c. Priority and FCFS
d. RR and SJF

Can you answer the following typical questions related to Operating Systems ?

Suppose that the following processes arrive for execution at the times indicated. Each process will run for the amount of time listed. In answering the questions, use nonpreemptive scheduling, and base all decisions on the information you have at the time the decision must be made.

Process	Arrival Time	Burst Time
P_1	0.0	8
P_2	0.4	4
P_3	1.0	1

a. What is the average turnaround time for these processes with the FCFS scheduling algorithm?

b. What is the average turnaround time for these processes with the SJF scheduling algorithm?

c. The SJF algorithm is supposed to improve performance, but notice that we chose to run process P_1 at time 0 because we did not know that two shorter processes would arrive soon. Compute what the average turnaround time will be if the CPU is left idle for the first 1 unit and then SJF scheduling is used. Remember that processes P_1 and P_2 are waiting during this idle time, so their waiting time may increase. This algorithm could be known as future-knowledge scheduling.

Can you answer the following typical questions related to Operating Systems ?

Suppose that a scheduling algorithm (at the level of short-term CPU scheduling) favors those processes that have used the least processor time in the recent past. Why will this algorithm favor I/O-bound programs and yet not permanently starve CPU-bound programs?

List three examples of deadlocks that are not related to a computer-system environment.

Suppose that a system is in an unsafe state. Show that it is possible for the processes to complete their execution without entering a deadlock state.

Can you answer the following typical questions related to Operating Systems ?

Consider the following snapshot of a system:

	Allocation	Max	Available
	A B C D	A B C D	A B C D
P_0	0 0 1 2	0 0 1 2	1 5 2 0
P_1	1 0 0 0	1 7 5 0	
P_2	1 3 5 4	2 3 5 6	
P_3	0 6 3 2	0 6 5 2	
P_4	0 0 1 4	0 6 5 6	

Answer the following questions using the banker's algorithm:

a. What is the content of the matrix **Need**?

b. Is the system in a safe state?

c. If a request from process P_1 arrives for (0,4,2,0), can the request be granted immediately?

Can you answer the following typical questions related to Operating Systems ?

Consider a computer system that runs 5,000 jobs per month with no deadlock-prevention or deadlock-avoidance scheme. Deadlocks occur about twice per month, and the operator must terminate and rerun about 10 jobs per deadlock. Each job is worth about $2 (in CPU time), and the jobs terminated tend to be about half-done when they are aborted.

A systems programmer has estimated that a deadlock-avoidance algorithm (like the banker's algorithm) could be installed in the system with an increase in the average execution time per job of about 10 percent. Since the machine currently has 30-percent idle time, all 5,000 jobs per month could still be run, although turnaround time would increase by about 20 percent on average.

a. What are the arguments for installing the deadlock-avoidance algorithm?

b. What are the arguments against installing the deadlock-avoidance algorithm?

Can you answer the following typical questions related to Operating Systems ?

Can a system detect that some of its processes are starving? If you answer "yes," explain how it can. If you answer "no," explain how the system can deal with the starvation problem.

Consider the following resource-allocation policy. Requests for and releases of resources are allowed at any time. If a request for resources cannot be satisfied because the resources are not available, then we check any processes that are blocked waiting for resources. If a blocked process has the desired resources, then these resources are taken away from it and are given to the requesting process. The vector of resources for which the blocked process is waiting is increased to include the resources that were taken away.

For example, consider a system with three resource types and the vector *Available* initialized to (4,2,2). If process P_0 asks for (2,2,1), it gets them. If P_1 asks for (1,0,1), it gets them. Then, if P_0 asks for (0,0,1), it is blocked (resource not available). If P_2 now asks for (2,0,0), it gets the available one (1,0,0) and one that was allocated to P_0 (since P_0 is blocked). P_0's *Allocation* vector goes down to (1,2,1), and its *Need* vector goes up to (1,0,1).

a. Can deadlock occur? If you answer "yes," give an example. If you answer "no," specify which necessary condition cannot occur.

b. Can indefinite blocking occur? Explain your answer.

Can you answer the following typical questions related to Operating Systems ?

Suppose that you have coded the deadlock-avoidance safety algorithm and now have been asked to implement the deadlock-detection algorithm. Can you do so by simply using the safety algorithm code and redefining **Max**[*i*] = **Waiting**[*i*] + **Allocation**[*i*], where **Waiting**[*i*] is a vector specifying the resources for which process *i* is waiting and **Allocation**[*i*] Explain your answer.

Is it possible to have a deadlock involving only one single-threaded process? Explain your answer.

Can you answer the following typical questions related to Operating Systems ?

Name two differences between logical and physical addresses.

Consider a system in which a program can be separated into two parts: code and data. The CPU knows whether it wants an instruction (instruction fetch) or data (data fetch or store). Therefore, two base–limit register pairs are provided: one for instructions and one for data. The instruction base–limit register pair is automatically read-only, so programs can be shared among different users. Discuss the advantages and disadvantages of this scheme.

Why are page sizes always powers of 2?

Consider a logical address space of 64 pages of 1024 words each, mapped onto a physical memory of 32 frames.

a. How many bits are there in the logical address?

b. How many bits are there in the physical address?

Can you answer the following typical questions related to Operating Systems ?

What is the effect of allowing two entries in a page table to point to the same page frame in memory? Explain how this effect could be used to decrease the amount of time needed to copy a large amount of memory from one place to another. What effect would updating some byte on the one page have on the other page?

Describe a mechanism by which one segment could belong to the address space of two different processes.

Sharing segments among processes without requiring that they have the same segment number is possible in a dynamically linked segmentation system.

a. Define a system that allows static linking and sharing of segments without requiring that the segment numbers be the same.

b. Describe a paging scheme that allows pages to be shared without requiring that the page numbers be the same.

Can you answer the following typical questions related to Operating Systems ?

Under what circumstances do page faults occur? Describe the actions taken by the operating system when a page fault occurs.

Assume that you have a page-reference string for a process with m frames (initially all empty). The page-reference string has length p; n distinct page numbers occur in it. Answer these questions for any page-replacement algorithms:

a. What is a lower bound on the number of page faults?

b. What is an upper bound on the number of page faults?

Consider the following page-replacement algorithms. Rank these algorithms on a five-point scale from "bad" to "perfect" according to their page-fault rate. Separate those algorithms that suffer from Belady's anomaly from those that do not.

a. LRU replacement

b. FIFO replacement

c. Optimal replacement

d. Second-chance replacement

Can you answer the following typical questions related to Operating Systems ?

Discuss the hardware support required to support demand paging.

Consider the two-dimensional array A:

```
int A[][] = new int[100][100];
```

where A[0][0] is at location 200 in a paged memory system with pages of size 200. A small process that manipulates the matrix resides in page 0 (locations 0 to 199). Thus, every instruction fetch will be from page 0.

For three page frames, how many page faults are generated by the following array-initialization loops, using LRU replacement and assuming that page frame 1 contains the process and the other two are initially empty?

a.
```
for (int j = 0; j < 100; j++)
    for (int i = 0; i < 100; i++)
        A[i][j] = 0;
```

b.
```
for (int i = 0; i < 100; i++)
    for (int j = 0; j < 100; j++)
        A[i][j] = 0;
```

Can you answer the following typical questions related to Operating Systems ?

Consider the following page reference string:

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6.

How many page faults would occur for the following replacement algorithms, assuming one, two, three, four, five, six, or seven frames? Remember all frames are initially empty, so your first unique pages will all cost one fault each.

- LRU replacement
- FIFO replacement
- Optimal replacement

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

Suppose that you want to use a paging algorithm that requires a reference bit (such as second-chance replacement or working-set model), but the hardware does not provide one. Sketch how you could simulate a reference bit even if one were not provided by the hardware, or explain why it is not possible to do so. If it is possible, calculate what the cost would be.

You have devised a new page-replacement algorithm that you think may be optimal. In some contorted test cases, Belady's anomaly occurs. Is the new algorithm optimal? Explain your answer.

Consider a demand-paged computer system where the degree of multiprogramming is currently fixed at four. The system was recently measured to determine utilization of CPU and the paging disk. The results are one of the following alternatives. For each case, what is happening? Can the degree of multiprogramming be increased to increase the CPU utilization? Is the paging helping?

a. CPU utilization 13 percent; disk utilization 97 percent

b. CPU utilization 87 percent; disk utilization 3 percent

c. CPU utilization 13 percent; disk utilization 3 percent

Can you answer the following typical questions related to Operating Systems ?

We have an operating system for a machine that uses base and limit registers, but we have modified the machine to provide a page table. Can the page tables be set up to simulate base and limit registers? How can they be, or why can they not be?

Consider the following set of processes, with the length of the CPU burst time given in milliseconds:

Process	Burst Time	Priority
P_1	2	2
P_2	1	1
P_3	8	4
P_4	4	2
P_5	5	3

The processes are assumed to have arrived in the order P_1, P_2, P_3, P_4, P_5 all at time 0.

a. Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: FCFS, SJF, nonpreemptive priority (a smaller priority number implies a higher priority), and RR (quantum = 1).
b. What is the turnaround time of each process for each of the scheduling algorithms in part a?
c. What is the waiting time of each process for each of these scheduling algorithms?
d. Which of the algorithms results in the minimum average waiting time (over all processes)?

Can you answer the following typical questions related to Operating Systems ?

The following processes are being scheduled using a preemptive, round-robin scheduling algorithm. Each process is assigned a numerical priority, with a higher number indicating a higher relative priority. In addition to the processes listed below, the system also has an **idle task** (which consumes no CPU resources and is identified as P_{idle}). This task has priority 0 and is scheduled whenever the system has no other available processes to run. The length of a time quantum is 10 units. If a process is preempted by a higher-priority process, the preempted process is placed at the end of the queue.

Thread	Priority	Burst	Arrival
P_1	40	20	0
P_2	30	25	25
P_3	30	25	30
P_4	35	15	60
P_5	5	10	100
P_6	10	10	105

a. Show the scheduling order of the processes using a Gantt chart.
b. What is the turnaround time for each process?
c. What is the waiting time for each process?
d. What is the CPU utilization rate?

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

Consider a preemptive priority scheduling algorithm based on dynamically changing priorities. Larger priority numbers imply higher priority. When a process is waiting for the CPU (in the ready queue, but not running), its priority changes at a rate α; when it is running, its priority changes at a rate β. All processes are given a priority of 0 when they enter the ready queue. The parameters α and β can be set to give many different scheduling algorithms.

a. What is the algorithm that results from $\beta > \alpha > 0$?

b. What is the algorithm that results from $\alpha < \beta < 0$?

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

- Describe the differences between symmetric and asymmetric multiprocessing. What are three advantages and one disadvantage of multiprocessor systems ?

- What is the purpose of interrupts ? What are the differences between a trap and an interrupt? Can traps be generated intentionally by a user program? If so, for what purpose ?

- Direct memory access is used for high-speed I/O devices in order to avoid increasing the CPU's execution load.
    - How does the CPU interface with the device to coordinate the transfer?
    - How does the CPU know when the memory operations are complete?
    - The CPU is allowed to execute other programs while the DMA controller is transferring data. Does this process interfere with the execution of the user programs? If so, describe what forms of interference are caused.

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

- Describe some of the challenges of designing operating systems for mobile devices compared with designing operating systems for traditional PCs.
- The services and functions provided by an operating system can be divided into two main categories. Briefly describe the two categories, and discuss how they differ.
- What are the two models of interprocess communication? What are the strengths and weaknesses of the two approaches?
- Why is the separation of mechanism and policy desirable?
- What is the main advantage of the microkernel approach to system design? How do user programs and system services interact in a microkernel architecture? What are the disadvantages of using the microkernel approach?
- Describe the differences among short-term, medium-term, and long-term scheduling
- Give an example of a situation in which ordinary pipes are more suitable than named pipes and an example of a situation in which named pipes are more suitable than ordinary pipes.

Can you answer the following typical questions related to Operating Systems ?

- What are the benefits and the disadvantages of each of the following ? Consider both the system level and the programmer level.
    - Synchronous and asynchronous communication
    - Automatic and explicit buffering
    - Send by copy and send by reference
    - Fixed-sized and variable-sized messages

- Why is it important for the scheduler to distinguish I/O-bound programs from CPU-bound programs ?

- One technique for implementing lottery scheduling works by assigning processes lottery tickets, which are used for allocating CPU time. Whenever a scheduling decision has to be made, a lottery ticket is chosen at random, and the process holding that ticket gets the CPU. The BTV operating system implements lottery scheduling by holding a lottery 50 times each second, with each lottery winner getting 20 milliseconds of CPU time (20 milliseconds × 50 = 1 second). Describe how the BTV scheduler can ensure that higher-priority threads receive more attention from the CPU than lower-priority threads.

Can you answer the following typical questions related to Operating Systems ?

- **Consider the exponential average formula used to predict the length of the next CPU burst. What are the implications of assigning the following values to the parameters used by the algorithm?**
  - **$\alpha = 0$ and $\tau_0 = 100$ milliseconds**
  - **$\alpha = 0.99$ and $\tau_0 = 10$ milliseconds**

- **When $\alpha = 0$ and $\tau_0 = 100$ mS, the formula always makes a prediction of 100 mS for the next CPU burst.**

- **When $\alpha = 0.99$ and $\tau_0 = 10$ mS, the most recent behavior of the process is given much higher weight than the past history associated with the process.**

- **Consequently, the scheduling algorithm is almost memoryless, and simply predicts the length of the previous burst for the next quantum of CPU execution.**

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

- **What is a bootstrap program, and where is it stored?**

- **It is the initial program that the computer runs when it is power up or rebooted. It initializes all aspects of the system.**

- **Typically it is stored in read-only memory (ROM) or electrically erasable programmable read-only memory (EEPROM), known by the general term firmware, within the computer hardware**

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

- **What role do device controllers and device drivers play in a computer system?**

- A general-purpose computer system consists of CPUs and multiple device controllers that are connected through a common bus.

- Each device controller is responsible for moving the data between the peripheral devices that it controls and its local buffer storage.

- Typically, operating systems have a device driver for each device controller. This device driver understands the device controller and presents a uniform interface for the device to the rest of the operating system.

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

- **Describe the differences between physical, virtual, and logical memory.**

- **Physical memory is the memory available for machines to execute operations (i.e., cache, random access memory, etc.).**

- **Virtual memory is a method through which programs can be executed that requires space larger than that available in physical memory by using disk memory as a backing store for main memory.**

- **Logical memory is an abstraction of the computer's different types of memory that allows programmers and applications a simplified view of memory and frees them from concern over memory-storage limitations.**

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

- **Describe the operating system's two modes of operation.**

- In order to ensure the proper execution of the operating system, most computer systems provide hardware support to distinguish between user mode and kernel mode.

- A mode bit is added to the hardware of the computer to indicate the current mode: kernel (0) or user (1).

- When the computer system is executing on behalf of a user application, the system is in user mode.

- However, when a user application requests a service from the operating system (via a system call), it must transition from user to kernel mode to fulfill the request

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

- **Why is main memory not suitable for permanent program storage or backup purposes? Furthermore, what is the main disadvantage to storing information on a magnetic disk drive as opposed to main memory?**

- **Main memory is a volatile memory in that any power loss to the system will result in erasure of the data stored within that memory. While disk drives can store more information permanently than main memory, disk drives are significantly slower**

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

- **Distinguish between system and application programs.**

- System programs are not part of the kernel, but still are associated with the operating system. Application programs are not associated with the operating of the system.

Can you answer the following typical questions related to Operating Systems ?

- **Describe why direct memory access (DMA) is considered an efficient mechanism for performing I/O.**

- **DMA is efficient for moving large amounts of data between I/O devices and main memory. It is considered efficient because it removes the CPU from being responsible for transferring data. DMA instructs the device controller to move data between the devices and main memory.**

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

- **Describe why multi-core processing is more efficient than placing each processor on its own chip.**

- **A large reason why it is more efficient is that communication between processors on the same chip is faster than processors on separate chips.**

Can you answer the following typical questions related to Operating Systems ?

- **Distinguish between uniform memory access (UMA) and non-uniform memory access (NUMA) systems**

- **On UMA systems, accessing RAM takes the same amount of time from any CPU. On NUMA systems, accessing some parts of memory may take longer than accessing other parts of memory, thus creating a performance penalty for certain memory accesses.**

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

- **Describe the relationship between an API, the system-call interface, and the operating system.**

- **The system-call interface of a programming language serves as a link to system calls made available by the operating system.**

- **This interface intercepts function calls in the API and invokes the necessary system call within the operating system.**

- **Thus, most of the details of the operating-system interface are hidden from the programmer by the API and are managed by the run-time support library.**

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

- **Describe some requirements, or goals, when designing an operating system.**
- **Requirements can be divided into user and system goals.**
- **Users desire a system that is convenient to use, easy to learn, and to use, reliable, safe, and fast.**
- **System goals are defined by those people who must design, create, maintain, and operate the system.**
- **The system should be easy to design, implement, and maintain; it should be flexible, reliable, error-free, and efficient.**

Can you answer the following typical questions related to Operating Systems ?

- **What are the advantages and disadvantages of using a microkernel approach?**

- One benefit of the microkernel approach is ease of extending the operating system.

- All new services are added to user space and consequently do not require modification of the kernel.

- The microkernel also provides more security and reliability, since most services are running as user — rather than kernel — processes.

- Unfortunately, microkernels can suffer from performance decreases due to increased system function overhead.

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

- **Explain why a modular kernel may be the best of the current operating system design techniques**

- **The modular approach combines the benefits of both the layered and microkernel design techniques.**

- **In a modular design, the kernel needs only to have the capability to perform the required functions and know how to communicate between modules.**

- **However, if more functionality is required in the kernel, then the user can dynamically load modules into the kernel.**

- **The kernel can have sections with well-defined, protected interfaces, a desirable property found in layered systems.**

- **More flexibility can be achieved by allowing the modules to communicate with one another.**

Can you answer the following typical questions related to Operating Systems ?

- **Name and describe the different states that a process can exist in at any given time.**

- **The possible states of a process are: new, running, waiting, ready, and terminated.**

- **The process is created while in the new state.**

- **In the running or waiting state, the process is executing or waiting for an event to occur, respectively.**

- **The ready state occurs when the process is ready and waiting to be assigned to a processor and should not be confused with the waiting state mentioned earlier.**

- **After the process is finished executing its code, it enters the termination state.**

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

- **Explain the concept of a context switch.**

- **Whenever the CPU starts executing a new process, the old process's state must be preserved.**

- **The context of a process is represented by its process control block. Switching the CPU to another process requires performing a state save of the current process and a state restore of a different process.**

- **This task is known as a context switch. When a context switch occurs, the kernel saves the context of the old process in its PCB and loads the saves context of the new process scheduled to run.**

Can you answer the following typical questions related to Operating Systems ?

- **List the four major categories of the benefits of multithreaded programming. Briefly explain each.**

- **The benefits of multithreaded programming fall into the categories: responsiveness, resource sharing, economy, and utilization of multiprocessor architectures.**

- **Responsiveness means that a multithreaded program can allow a program to run even if part of it is blocked.**

- **Resource sharing occurs when an application has several different threads of activity within the same address space.**

- **Threads share the resources of the process to which they belong. As a result, it is more economical to create new threads than new processes.**

- **Finally, a single-threaded process can only execute on one processor regardless of the number of processors actually present. Multiple threads can run on multiple processors, thereby increasing efficiency.**

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

- **What are the two different ways in which a thread library could be implemented?**

- **The first technique of implementing the library involves ensuring that all code and data structures for the library reside in user space with no kernel support.**

- **The other approach is to implement a kernel-level library supported directly by the operating system so that the code and data structures exist in kernel space**

Can you answer the following typical questions related to Operating Systems ?

- **What is a thread pool and why is it used?**

- **A thread pool is a collection of threads, created at process startup, that sit and wait for work to be allocated to them.**

- **This allows one to place a bound on the number of concurrent threads associated with a process and reduce the overhead of creating new threads and destroying them at termination.**

Can you answer the following typical questions related to Operating Systems ?

- **Explain the process of starvation and how aging can be used to prevent it.**

- **Starvation occurs when a process is ready to run but is stuck waiting indefinitely for the CPU.**

- **This can be caused, for example, when higher-priority processes prevent low-priority processes from ever getting the CPU.**

- **Aging involves gradually increasing the priority of a process so that a process will eventually achieve a high enough priority to execute if it waited for a long enough period of time.**

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

- **Explain two general approaches to handle critical sections in operating systems.**
- **Critical sections may use preemptive or nonpreemptive kernels.**
- **A preemptive kernel allows a process to be preempted while it is running in kernel mode.**
- **A nonpreemptive kernel does not allow a process running in kernel mode to be preempted; a kernel-mode process will run until it exits kernel mode, blocks, or voluntarily yields control of the CPU.**
- **A nonpreemptive kernel is essentially free from race conditions on kernel data structures, as the contents of this register will be saved and restored by the interrupt handler.**

Can you answer the following typical questions related to Operating Systems ?

- **Write two short functions that implement the simple semaphore wait() and signal() operations on global variable S.**

```
wait (S) {
while (S <= 0);
S--;
}

signal (S) {
S++;
}
```

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

- **Describe the dining-philosophers problem and how it relates to operating systems.**

- **The scenario involves five philosophers sitting at a round table with a bowl of food and five chopsticks.**

- **Each chopstick sits between two adjacent philosophers.**

- **The philosophers are allowed to think and eat. Since two chopsticks are required for each philosopher to eat, and only five chopsticks exist at the table, no two adjacent philosophers may be eating at the same time.**

- **A scheduling problem arises as to who gets to eat at what time. This problem is similar to the problem of scheduling processes that require a limited number of resources.**

THANK YOU

Nitin V Pujari
Faculty, Computer Science
Dean, IQAC, PES University

nitin.pujari@pes.edu

For Course Deliverables by the Anchor Faculty click on www.pesuacademy.com