CS Exit Ethiopia

Questions & Worked Answers

Computer Science Exit Examination

233 complete questions and 1 source variants

1 selected course | 18 November 2026

Selected courses

01. Operating System

https://csexitethiopia.com

abelasnake2003@gmail.com | Telegram: @Abechu7

Course contents

Each course begins on a new page. PDF bookmarks provide question, study-note section and revision-topic navigation.

Operating System 3

Reading the answers

Every question includes its answer, explanation, choice-by-choice contrasts and available evidence. Primary note proof quotes the study notes. Related reading and module passages are labelled separately; a module reference is not presented as primary note proof.

Operating System

Questions & Worked Answers

233 questions + 1 source variants

Question 1 | Q00023

Which mutual exclusion mechanism keeps track of turns to enter the critical region using an integer variable, which is initially-0, and another process waits until it turns to 1?

A. Lock variable

B. Disabling interrupts

C. Strict alteration

D. Race condition

Hint

Track who may enter and distinguish separate tests and updates from atomic operations.

Answer C: Strict alternation (option misspells it as alteration).

OS12 defines strict alternation using turn=0/1: each process waits until turn equals its identifier and passes the turn on leaving. That directly matches the described integer ownership variable. A naive lock variable instead records availability and can race during separate test/set operations. Disabling interrupts and a race condition do not implement the described turn-taking algorithm.

Why each choice is right or wrong

A - Incorrect: A naive lock variable records free/busy status rather than assigning alternating ownership to process 0 or 1.

B - Incorrect: Disabling interrupts does not implement the integer turn-owner protocol described in the stem.

C - Correct: Strict alternation uses a turn variable initialized to one process and passed to the other.

D - Incorrect: A race condition is a concurrency failure, not a mechanism providing the described alternating access.

Primary note proof

Study-note section OS12

> Strict alternation uses turn=0/1: process i waits while turn≠i, then enters and sets turn to the other process on leaving. It enforces turns but can violate progress if the other process stays outside its critical region. A naive lock-variable test followed by a separate set races because both processes can observe free before either sets it.

Study-note section OS12

Test-and-set and compare-and-swap combine checking/updating atomically; ordinary check-then-write does not. Peterson’s classic algorithm uses flags plus turn for two processes under its memory-model assumptions; a semaphore can coordinate more than two. Disabling local interrupts does not exclude other CPUs and is not a general user-space locking solution.

Study-note section OS10

A race occurs when results depend on uncoordinated interleaving of conflicting accesses.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

• Loads a program into memory.
• Executes the program.
• Handles program's execution.
• Provides a mechanism for process synchronization.
• Provides a mechanism for process communication.
• Provides a mechanism for deadlock handling.

Operating System module | PDF page 45 | lines 10-22

The CPU hardware uses an interrupt request line wire which helps CPU to sense after executing every instruction. When the CPU checks that a controller has put a signal on the interrupt request line, the CPU saves a state, such as the current value of the instruction pointer, and jumps to the interrupt handler routine at a fixed address. The interrupt handler part determines the cause of the interrupt performs the necessary processing and executes an interrupt instruction to return the CPU to its execution state.
The basic mechanism of interrupt enables the CPU to respond to an asynchronous event, such as when a device controller becomes ready for service. Most CPUs have two interrupt request lines.
• non-maskable interrupt - Such kind of interrupts are reserved for events like unrecoverable memory errors.
• maskable interrupt - Such interrupts can be switched off by the CPU before the execution of critical instructions that must not be interrupted.

Question sources

MOE Computer Science Exit Exam · 2016 · Question 23

Question 2 | Q00031

Which one of the following messaging communication best explains the statement: "Until the message is received, the sending process is blocked'"?

A. Blocking receive

B. Non-block receive

C. Non-block sending

D. Blocking send

Hint

Identify how data moves, which participant waits and what event allows the operation to continue.

Answer D: Blocking send.

OS09 explicitly identifies a sender blocked until receipt as blocking send under rendezvous semantics. The stem describes the sending process waiting, not a receiver awaiting an incoming message. Nonblocking sending returns promptly rather than waiting for receipt, and receive modes describe the other side. The receipt condition here supplies the relevant API assumption.

Why each choice is right or wrong

A - Incorrect: Blocking receive waits on the receiving side, whereas the stem explicitly blocks the sender.

B - Incorrect: Nonblocking receive returns promptly on the receiving side and does not describe sender waiting.

C - Incorrect: Nonblocking sending returns promptly rather than keeping the sender blocked until the handoff completes.

D - Correct: Under the stated rendezvous condition, waiting until message receipt is blocking-send behavior.

Primary note proof

Study-note section OS09

> Blocking send waits for the required handoff/buffering/receipt condition defined by its API. Blocking receive waits for a message. Nonblocking operations return promptly with data, progress or a would-block result. “Sender blocked until received” identifies blocking send under rendezvous semantics.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 11 | lines 15-19

> • Two processes often require data to be transferred between them.
> • The both processes can be on the one computer or on different computer but are connected through computer network.
> • Communication may be implemented by two methods either by Shared Memory or by Message Passing.

Question sources

MOE Computer Science Exit Exam · 2016 · Question 31

Question 3 | Q00034

_____is the component of the operating system that is responsible to decide which process to run first among ready processes.

A. Memory manager

B. Inter process communication

C. Disk manager

D. Scheduler

Hint

Decide whether the process is eligible, running or waiting, then identify the responsible queue or scheduling role.

Answer D: Scheduler

The CPU scheduler selects among ready processes and allocates the CPU (PDF p20). This is precisely the selection task in the stem.

Why each choice is right or wrong

A - Incorrect: Memory allocation is a separate function (p5).

B - Incorrect: Process data exchange (p11) is not selection of the next CPU user.

C - Incorrect: Device allocation (p6) does not decide CPU order.

D - Correct: CPU scheduling selects ready processes (p20).

Primary note proof

Study-note section OS06

> The ready queue contains eligible work; device queues hold blocked requests. The broad textbook job queue contains processes in the system, while an admission queue more narrowly holds jobs awaiting memory/admission. Long-term scheduling admits jobs; medium-term scheduling suspends/swaps them; short-term scheduling selects ready work. The dispatcher performs the handoff/context restore after selection. A bare “priority” causes running→ready only if a preemptive policy chooses another process.

Study-note section OS01

> An OS allocates CPU time, memory, storage, devices and communication resources, and provides services to programs.

Study-note section OS09

> Shared memory lets processes access a common region and requires synchronization for conflicting updates. Message passing exchanges data through send/receive operations.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 20 | lines 15-21

It is also called CPU scheduler. Main objective is increasing system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. CPU scheduler selects process among the processes that are ready to execute and allocates CPU to one of them.
Short term scheduler also known as dispatcher, execute most frequently and makes the fine-grained decision of which process to execute next. Short term scheduler is faster than long term scheduler.

Operating System module | PDF page 5 | lines 26-36

Memory management refers to management of Primary Memory or Main Memory. Main memory is a large array of words or bytes where each word or byte has its own address. Main memory provides a fast storage that can be access directly by the CPU. So, for a program to be executed, must in the main memory.
Activities Operating System for memory management.
Keeps tracks of primary memory (i.e. what part of it are in use by whom, what part are not in use.)
In multiprogramming, OS decides which process will get memory when and how much.
Allocates the memory when the process requests it to do so.
De-allocates the memory when the process no longer needs it or has been terminated.

Operating System module | PDF page 11 | lines 15-19

• Two processes often require data to be transferred between them.
• The both processes can be on the one computer or on different computer but are connected through computer network.
• Communication may be implemented by two methods either by Shared Memory or by Message Passing.

Operating System module | PDF page 6 | lines 11-17

OS manages device communication via their respective drivers. Activities of Operating System for device management.
Keeps tracks of all devices. Program responsible for this task is known as the I/O controller.
Decides which process gets the device when and for how much time.
Allocates the device in the efficient way.
De-allocates devices.

Question sources

MOE Computer Science Exit Exam · 2016 · Question 34

Question 4 | Q00092

Which one of the following is not true?

A. kernel is the program that constitutes the central core of the operating system

B. kernel is the first part of operating system to load into memory during booting

C. kernel is made of various modules which cannot be loaded in running operating system

D. kernel remains in the memory during the entire computer session

Hint

Distinguish command interfaces, privileged core services and the stages of booting.

Answer C: The claim that kernel modules cannot load while running is false.

OS03 states that loadable modules can add drivers to a running kernel, directly contradicting option C's absolute prohibition. It describes the kernel as the core OS component and resident through the session. Option B is read as the first core OS component, rather than the first code executed at power-on: firmware and bootloader stages precede it.

Why each choice is right or wrong

A - Incorrect: The kernel implements core resource management, protection and system-call handling, so this statement is true.

B - Incorrect: As the first core OS component, the kernel is loaded at boot; firmware/bootloader execution precedes that OS boundary.

C - Correct: Modular kernels permit runtime module loading, directly refuting the statement that it cannot occur.

D - Incorrect: A resident kernel remains active throughout the session, matching the note's ordinary kernel model.

Primary note proof

Study-note section OS03

> The kernel implements core resource management, protection and system-call handling.

Study-note section OS03

> Firmware/bootloader stages execute before transferring control to a kernel loaded into RAM. The kernel is the first core OS component, not the first program ever executed at power-on. Loadable modules can add drivers to a running kernel; "kernel modules cannot be loaded while running" is false in modular systems. A resident kernel remains active through the session. Utilities such as antivirus and disk tools can combine user processes with privileged components.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 53 | lines 11-15

operating system. It is consisting of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.
System Library - System libraries are special functions or programs using which application programs or system utilities accesses Kernel's features. These libraries

Operating System module | PDF page 32 | lines 11-14

In dynamic loading, a routine of a program is not loaded until it is called by the program. All routines are kept on disk in a re-locatable load format. The main program is loaded into memory and is executed. Other routines methods or modules are loaded on request. Dynamic loading makes better memory space utilization and unused routines are never loaded.

Question sources

MOE Computer Science Exit Exam · 2016 · Question 92

Computer Science model exam · Question 95

University of Gondar model exam · Question 95

Question 5 | Q00164

One is not the goal of I/O software

A. Buffering

B. Device independence

C. Uniform naming

D. Error handling

E. None of the above

Hint

Separate device hardware, device-specific software and general event or transfer mechanisms.

Answer E: None: all four are I/O-software goals or supporting functions.

OS34 identifies device independence, uniform naming and error handling as I/O services. OS35 explains buffering as holding in-transit data to absorb speed/size differences. Each listed feature therefore supports I/O software, so none is an exclusion. Calling buffering a supporting mechanism rather than a high-level objective does not make it unrelated to I/O software’s goals.

Why each choice is right or wrong

A - Incorrect: Buffering absorbs transfer-speed and size differences, directly supporting reliable and efficient I/O software.

B - Incorrect: Device independence is an explicit I/O service that hides device-specific details from higher layers.

C - Incorrect: Uniform naming is explicitly included among the I/O services described here.

D - Incorrect: Reporting or retrying supported device failures is the error-handling responsibility of I/O software.

E - Correct: All four listed functions support I/O software, leaving no listed function to exclude.

Primary note proof

Study-note section OS34

> A driver converts OS requests into device-specific operations. Device independence and uniform naming hide brand-specific details; error handling reports/retries supported failures.

Study-note section OS35

> Buffering temporarily holds data in transit to absorb speed/transfer-size differences.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 45 | lines 28-33

> Application I/O Interface represents the structuring techniques and interfaces for the operating system to enable I/O devices to be treated in a standard, uniform way. The actual differences lies kernel level modules called device drivers which are custom tailored to corresponding devices but show one of the standard interfaces to applications. The purpose of the device-driver layer is to hide the differences among device controllers from the I/O subsystem of the kernel, such as the I/O system calls. Following are the characteristics of I/O

Operating System module | PDF page 47 | lines 5-9

> Buffering - Kernel I/O Subsystem maintains a memory area known as buffer that stores data while they are transferred between two devices or between a devices with an application operation. Buffering is done to cope with a speed mismatch between the producer and consumer of a data stream or to adapt between devices that have different data transfer sizes.

Operating System module | PDF page 47 | lines 17-18

> Error Handling - An operating system that uses protected memory can guard against many kinds of hardware and application errors.

Question sources

Addis Ababa University mock exam · Question 66

Question 6 | Q00165

Which hardware unit translates a process's logical/virtual addresses to physical memory addresses during execution?

A. MMU

B. IRQ

C. DMA

D. Cache

E. Relocation Register

Hint

Track when an address or library reference is resolved and the direction of translation.

Answer A: MMU

The MMU performs execution-time translation from a process's logical or virtual address to a physical RAM address. A relocation register can supply a base used in a simple translation, but it is a register rather than the complete translating unit. IRQ concerns interrupts, DMA moves data, and a cache stores reusable information. The corrected direction therefore identifies the MMU uniquely.

Why each choice is right or wrong

A - Correct: The MMU is the hardware unit performing logical-to-physical translation.

B - Incorrect: An IRQ signals an interrupt rather than translating process addresses.

C - Incorrect: DMA transfers data between devices and memory after setup.

D - Incorrect: A cache retains reusable contents; it is not the general address-translating unit.

E - Incorrect: A relocation register holds a base parameter used by translation hardware.

Primary note proof

Study-note section OS27

> Logical/virtual addresses are generated for a process; physical addresses select RAM. An MMU translates logical addresses to physical addresses, not the reverse.

Study-note section OS28

> In the standard base/limit model, base is that process's physical starting address and limit is its permitted length. Check 0≤logical_offset<limit, then physical=base+offset.

Study-note section OS34

> Polling checks readiness repeatedly; interrupts request CPU attention; DMA transfers blocks after setup without CPU work for every byte.

Study-note section OS35

> Caching retains copies to accelerate reuse.

Study-note section OS-S04

IRQ means interrupt request.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 33 | lines 6-13

The run-time mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device. MMU uses following mechanism to convert virtual address to physical address.
• The value in the base register is added to every address generated by a user process which is treated as offset at the time it is sent to memory. For example, if the base register value is 10000, then an attempt by the user to use address location 100 will be dynamically reallocated to location 10100.
• The user program deals with virtual addresses; it never sees the real physical addresses.

Operating System module | PDF page 34 | lines 12-16

In this type of allocation, relocation-register scheme is used to protect user processes from each other, and from changing operating-system code and data. Relocation register contains value of smallest physical address whereas limit register contains range of logical addresses. Each logical address must be less than the limit register.

Operating System module | PDF page 45 | lines 10-22

The CPU hardware uses an interrupt request line wire which helps CPU to sense after executing every instruction. When the CPU checks that a controller has put a signal on the interrupt request line, the CPU saves a state, such as the current value of the instruction pointer, and jumps to the interrupt handler routine at a fixed address. The interrupt handler part determines the cause of the interrupt performs the necessary processing and executes an interrupt instruction to return the CPU to its execution state.
The basic mechanism of interrupt enables the CPU to respond to an asynchronous event, such as when a device controller becomes ready for service. Most CPUs have two interrupt request lines.
• non-maskable interrupt - Such kind of interrupts are reserved for events like unrecoverable memory errors.
• maskable interrupt - Such interrupts can be switched off by the CPU before the execution of critical instructions that must not be interrupted.

Operating System module | PDF page 47 | lines 10-11

Caching - Kernel maintains cache memory which is region of fast memory that holds copies of data. Access to the cached copy is more efficient than access to the original.

Question sources

Addis Ababa University mock exam · Question 67

Question 7 | Q00166

For holes of sizes 100, 500, 200 and 300 in that scan order and a request of size 180, which policy is better at minimizing this request's leftover space in its selected hole? Ignore search cost and future requests.

A. First fit algorithm is better

B. Best fit algorithm is better

C. Worst fit algorithm is better

D. All of the above

Hint

Apply the stated hole-selection or base-and-bound rule with its explicit convention.

Answer B: Best fit algorithm is better

Best-fit chooses the smallest hole large enough for the request: 200, leaving 20 unused units. First-fit stops at 500 and leaves 320 units; worst-fit also chooses 500. Under the stated objective, best-fit gives the smallest remainder for this allocation. This calculation says nothing about future fragmentation or overall search speed, which require a broader workload.

Why each choice is right or wrong

A - Incorrect: First-fit selects 500 in this scan order, leaving 320 rather than the minimum remainder.

B - Correct: Best-fit selects 200 and leaves only 20 units.

C - Incorrect: Worst-fit selects 500 and leaves 320 units.

D - Incorrect: The policies do not tie: only best-fit minimizes this request's leftover space.

Primary note proof

Study-note section OS28

> First-fit selects the first adequate hole; next-fit continues from the previous search position; best-fit selects the smallest adequate hole; worst-fit selects the largest.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 34 | lines 34-45

> External
> fragmentation
> Total memory space is enough to satisfy a request or to reside a
> process in it, but it is not contiguous so it cannot be used.
> 2
> Internal
> fragmentation
> Memory block assigned to process is bigger. Some portion of memory

is left unused as it cannot be used by another process.
External fragmentation can be reduced by compaction or shuffle memory contents to place all free memory together in one large block. To make compaction feasible, relocation should be dynamic

Question sources

Addis Ababa University mock exam · Question 68

Question 8 | Q00167

All jobs arrive together at submission, have known finite CPU bursts, perform no I/O, and no further jobs arrive. Assume one CPU, zero switching overhead, known fixed priorities (smaller value is higher; non-preemptive priority), a known positive RR quantum, and listed order for ties. For which listed algorithms can each job's total waiting time be computed at submission? Choose the most complete answer.

A. Shortest job first

B. Shortest remaining time

C. Priority based

D. Round Robin

E. All of the above

Hint

Construct a timeline and distinguish completion rates from individual elapsed-time measures.

Answer E: All of the above

The closed workload and explicit policy parameters determine each scheduling timeline. SJF orders known bursts, SRTF uses remaining bursts, priority uses the supplied fixed order, and RR uses its quantum and ready queue. Completion minus arrival minus burst then gives waiting time. Each named algorithm is individually computable, so the aggregate option is the most complete answer under these assumptions.

Why each choice is right or wrong

A - Incorrect: SJF waiting is computable, but this option omits the other computable policies.

B - Incorrect: SRTF waiting is computable for the known closed workload, but this is incomplete.

C - Incorrect: Fixed-priority waiting is computable with the supplied priorities and tie rule, but this is incomplete.

D - Incorrect: RR waiting is computable with the known quantum and queue order, but this is incomplete.

E - Correct: All four schedules and their total waiting times can be computed from the fully specified finite workload.

Primary note proof

Study-note section OS13

For arrival A, first start S and completion C, response=S−A and turnaround=C−A. For a single CPU burst B with no I/O, waiting=C−A−B.

Study-note section OS16

Non-preemptive SJF/SJN chooses the shortest next burst among currently ready jobs. SRTF reconsiders on arrivals and selects the shortest remaining work; a shorter newcomer can preempt.

Study-note section OS17

Priority scheduling chooses the highest-priority ready job. Check whether a smaller or larger numeric value means higher priority. State whether a high-priority arrival preempts.

Study-note section OS18

RR serves a ready queue cyclically for at most quantum q per turn. Unfinished jobs return to the tail; completed/blocked jobs leave.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 23 | lines 23-26

7.2 Shortest Job First (SJF)
• Best approach to minimize waiting time.
• Impossible to implement
• Processer should know in advance how much time process will take.

Operating System module | PDF page 24 | lines 10-15

7.3 Priority Based Scheduling
• Each process is assigned a priority. Process with highest priority is to be executed first and so on.
• Processes with same priority are executed on first come first serve basis.
• Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Operating System module | PDF page 25 | lines 10-14

7.4 Round Robin Scheduling
• Each process is provided a fix time to execute called quantum.
• Once a process is executed for given time period. Process is preempted and other process executes for given time period.
• Context switching is used to save states of preempted processes.

Question sources

Addis Ababa University mock exam · Question 69

Question 9 | Q00168

One is not a problem caused by concurrency of processes

A. Race condition

B. Starvation

C. Deadlock

D. Mutual exclusion

E. None

Hint

Identify conflicting shared accesses and the safety or progress property required.

Answer D: Mutual exclusion is a desired safety property.

OS10 explicitly contrasts mutual exclusion, a desired safety property, with race conditions, starvation and deadlock as possible failures. Although a badly designed locking scheme can introduce deadlock, that does not turn the mutual-exclusion requirement itself into a concurrency failure. Therefore it is the single option that is not a problem in the sense used by the list.

Why each choice is right or wrong

A - Incorrect: A race is a concurrency failure caused by uncoordinated interleaving of conflicting accesses.

B - Incorrect: Starvation is a possible indefinite-delay failure of concurrent scheduling or resource competition.

C - Incorrect: Deadlock is a possible failure in which blocked participants cannot supply one another's required events.

D - Correct: Mutual exclusion is a correctness property protecting shared state, rather than a failure to be avoided.

E - Incorrect: Mutual exclusion provides a clear non-failure option, so none of the choices is not appropriate.

Primary note proof

Study-note section OS10

> A race occurs when results depend on uncoordinated interleaving of conflicting accesses. A critical section is code accessing shared state that needs protection, not the resource itself. A correct exclusion solution requires mutual exclusion, progress and bounded waiting under stated assumptions. Mutual exclusion is a desired safety property; races, starvation and deadlock are possible failures.

Study-note section OS22

A set is deadlocked when each member waits for an event/resource that only another blocked member can provide. Mere waiting for another holder is normal blocking; indefinite unfair delay can be starvation without a cycle of dependencies.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

- Loads a program into memory.
- Executes the program.
- Handles program's execution.
- Provides a mechanism for process synchronization.
- Provides a mechanism for process communication.
- Provides a mechanism for deadlock handling.

Question sources

Addis Ababa University mock exam · Question 70

Question 10 | Q00169

Which methods can participate in recovery after deadlock has been detected? Assume checkpoints and safe resource reclamation are available where needed. Select all that apply.

A. Kill the process

B. Rollback

C. Non-preempt the resource

D. Lock the process

Hint

Separate handling before allocation from actions taken after detection, and consider the cost of checking.

Answer A, B: Kill the process; Rollback

Terminating a selected process can release resources and break the dependency cycle. Recovery may instead reclaim safely preemptable resources and restore affected work to a checkpoint, so rollback can also participate. Merely preserving non-preemption or locking a process does not remove the deadlocked dependencies. With checkpoint and reclamation support stated, both termination and rollback are valid recovery tools.

Why each choice is right or wrong

A - Correct: Killing a selected victim can release resources and break the deadlock.

B - Correct: Rollback can restore affected work after safe resource reclamation.

C - Incorrect: Non-preemption preserves a deadlock-enabling restriction rather than recovering the processes.

D - Incorrect: Locking a blocked process does not release resources or restore progress.

Primary note proof

Study-note section OS26

> Detection asks whether the current state is deadlocked. Recovery can terminate selected processes or reclaim safely preemptable resources and roll affected work back to a checkpoint. Termination and rollback are both possible recovery tools; choose according to the resource and process conditions.

Study-note section OS22

> No resource preemption means release must be voluntary.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

> • Loads a program into memory.
> • Executes the program.
> • Handles program's execution.
> • Provides a mechanism for process synchronization.
> • Provides a mechanism for process communication.
> • Provides a mechanism for deadlock handling.

Question sources

Addis Ababa University mock exam · Question 71

Question 11 | Q00170

One deadlock prevention method says, "if a process must wait for a needed resource, it drops all of its previously held resources and tries to acquire all resources gain." These attacks

A. Circular wait condition

B. No preemption conditions

C. Mutual exclusion condition

D. Hold and wait condition

E. None of the above

Hint

Identify the resource-request restriction and which necessary condition it removes.

Answer D: The policy removes hold-and-wait.

Under this policy, a process releases all previously held resources when it must wait. Releasing all resources before retrying removes simultaneous holding while waiting. The stem describes voluntary dropping, not forced withdrawal by another agent, so hold-and-wait is the directly attacked condition. It supplies no global ordering or resource-sharing rule.

Why each choice is right or wrong

A - Incorrect: The policy does not impose a global acquisition order, which is the direct circular-wait prevention method.

B - Incorrect: The stem specifies the process dropping resources, not a general rule permitting forced resource preemption.

C - Incorrect: Resources may remain nonshareable; releasing holdings does not remove mutual-exclusion requirements.

D - Correct: A process no longer retains allocated resources while waiting to acquire the rest.

E - Incorrect: Hold-and-wait is explicitly removed by the release-all-before-waiting rule, so an identified condition fits.

Primary note proof

Study-note section OS24

> Prevention constrains how requests are made. Share a resource where safe to remove mutual exclusion; request all needed resources before holding any to remove hold-and-wait; permit safe resource withdrawal/restart to attack no-preemption; impose a global acquisition order to remove circular wait. Not every physical resource is shareable or safely preemptable.

Study-note section OS24

> “If waiting, release all held resources and try again” removes simultaneous holding while waiting.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

> • Loads a program into memory.
> • Executes the program.
> • Handles program's execution.
> • Provides a mechanism for process synchronization.
> • Provides a mechanism for process communication.
> • Provides a mechanism for deadlock handling.

Question sources

Addis Ababa University mock exam · Question 72

Question 12 | Q00229

Which statements are false? Interpret aging under a priority scheme and workload where increasing waiting jobs' priority prevents indefinite postponement. Select all that apply.

A. File system standards of various OS is the same.

B. SJF may cause starvation.

C. Aging is the solution of the starvation problem in the case of priority scheduling.

D. Critical section is a resource that cannot be accessed by multiple processes concurrently.

Hint

Identify conflicting shared accesses and the safety or progress property required.

Answer A, D: File system standards of various OS is the same.; Critical section is a resource that cannot be accessed by multiple processes concurrently.

Filesystems are not governed by one identical format across all operating systems; the notes distinguish FAT32, NTFS and ext4. A critical section is code that accesses shared state, not the resource itself. SJF can starve a long job when short work continues arriving. Aging addresses priority starvation under the stated policy assumptions, so those latter two statements are true.

Why each choice is right or wrong

A - Correct: The universal sameness claim is false; different filesystem formats exist.

B - Incorrect: SJF may starve long work under repeated short arrivals, so this statement is true.

C - Incorrect: Aging raises effective priority and addresses starvation under the specified conditions, so this statement is true.

D - Correct: A critical section is protected code rather than the shared resource, making this statement false.

Primary note proof

Study-note section OS10

> A critical section is code accessing shared state that needs protection, not the resource itself.

Study-note section OS17

> Fixed-priority scheduling can starve low-priority jobs under continued high-priority load. SJF can also starve long jobs; starvation is not exclusive to fixed-priority scheduling.

Study-note section OS17

> Aging increases waiting jobs' effective priority according to a rule, reducing indefinite postponement when the priority scheme and workload permit.

Study-note section OS36

> Sockets also support ordinary read/write, alongside socket/connect/send operations, so read is not uniquely "not a network call." FAT32, NTFS and ext4 are filesystem formats; HTTP is an application communication protocol.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 48 | lines 5-8

13.1 File
A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks. In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the file's creator and user.

Operating System module | PDF page 24 | lines 10-15

7.3 Priority Based Scheduling
• Each process is assigned a priority. Process with highest priority is to be executed first and so on.
• Processes with same priority are executed on first come first serve basis.
• Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Operating System module | PDF page 10 | lines 23-28

• Loads a program into memory.
• Executes the program.
• Handles program's execution.
• Provides a mechanism for process synchronization.
• Provides a mechanism for process communication.
• Provides a mechanism for deadlock handling.

Question sources

Mock exam 4 · Question 31

Question 13 | Q00230

One of the following system calls is used to create a new process.

A. Fork()

B. exit()

C. abort()

D. exec()

Hint

Separate creation, image replacement and termination; classify termination by its cause.

Answer A: fork().

OS07 states that fork() creates a child in POSIX-style systems, returning zero in the child and its PID in the parent on success. exec replaces the current process image and does not itself create another process. exit and abort terminate execution normally and abnormally respectively. Creation therefore corresponds to fork among the literal listed calls.

Why each choice is right or wrong

A - Correct: fork creates a child process rather than replacing or terminating the existing process image.

B - Incorrect: exit terminates normally through the runtime rather than creating a new process.

C - Incorrect: abort requests abnormal termination rather than allocating a new child execution instance.

D - Incorrect: exec replaces the current process image; it does not itself create another process.

Primary note proof

Study-note section OS07

> In POSIX-style systems fork() creates a child: zero is returned in the child and the child PID in the parent on success. The two have distinct address spaces, although resources may be shared through explicit mechanisms. exec replaces the current process image; it does not itself create another process.

Study-note section OS07

> Normal completion and a program-detected error exit are voluntary; an unrecoverable fault or being killed is involuntary. exit() terminates normally through the runtime; abort() requests abnormal termination.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 19 | lines 6-12

> A newly arrived process is put in the ready queue. Processes waits in ready queue for allocating the CPU. Once the CPU is assigned to a process, then that process will execute. While executing the process, any one of the following events can occur.
> • The process could issue an I/O request and then it would be placed in an I/O queue.
> • The process could create new sub process and will wait for its termination.
> • The process could be removed forcibly from the CPU, as a result of interrupt and put back in the ready queue.

Operating System module | PDF page 16 | lines 21-24

> A program by itself is not a process. It is a static entity made up of program statement while process is a dynamic entity. Program contains the instructions to be executed by processor. A program takes a space at single place in main memory and continues to stay there. A program does not perform any action by itself.

Question sources

Mock exam 4 · Question 32

Question 14 | Q00232

Which of the following process scheduling algorithms select the next process to be executed based on each process burst time?

A. FCFS

B. Shortest Job First

C. Round robin

D. Priority scheduling algorithm

Hint

Compare the ready jobs using the stated burst or remaining-time rule.

Answer B: Shortest Job First

SJF requires knowing how long a process will take (PDF p23); that is the burst-length selection described. Priority may use time as one possible priority input (p24), but its general rule is highest assigned priority.

Why each choice is right or wrong

A - Incorrect: FCFS orders arrival (p23), not burst length.

B - Correct: SJF chooses short work based on execution duration (p23).

C - Incorrect: RR assigns a fixed quantum (p25).

D - Incorrect: Priority selects assigned priority; time requirements are only one possible input (p24).

Primary note proof

Study-note section OS16

> Non-preemptive SJF/SJN chooses the shortest next burst among currently ready jobs.

Study-note section OS15

> FCFS runs ready bursts in arrival order and ordinarily does not preempt.

Study-note section OS17

> Priority scheduling chooses the highest-priority ready job.

Study-note section OS18

> RR serves a ready queue cyclically for at most quantum q per turn.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 23 | lines 23-26

> 7.2 Shortest Job First (SJF)
> • Best approach to minimize waiting time.

• Impossible to implement
• Processer should know in advance how much time process will take.

Operating System module | PDF page 23 | lines 12-17

7.1 First Come First Serve (FCFS)
• Jobs are executed on first come, first serve basis.
• Easy to understand and implement.
• Poor in performance as average wait time is high.
Wait time of each process is following
Process Wait Time: Service Time - Arrival Time

Operating System module | PDF page 25 | lines 10-14

7.4 Round Robin Scheduling
• Each process is provided a fix time to execute called quantum.
• Once a process is executed for given time period. Process is preempted and other process executes for given time period.
• Context switching is used to save states of preempted processes.

Operating System module | PDF page 24 | lines 10-15

7.3 Priority Based Scheduling
• Each process is assigned a priority. Process with highest priority is to be executed first and so on.
• Processes with same priority are executed on first come first serve basis.
• Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Question sources

Mock exam 4 · Question 34

Question 15 | Q00233

_____ is a software solution of a critical section problem if the number of instances and processes are more than two.

A. Semaphore

B. Compare and swap

C. Peterson

D. Test set

Hint

Track available permits and blocked operations using the stated semaphore model.

Answer A: Semaphore.

OS12 explicitly says a semaphore can coordinate more than two processes, while classic Peterson's algorithm is for two under its memory-model assumptions. OS11 explains counting semaphores as tracking several permits, fitting multiple resource instances. Compare-and-swap and test-and-set are atomic primitives rather than the named software coordination abstraction requested by the stem.

Why each choice is right or wrong

A - Correct: Counting semaphores track multiple permits and coordinate more than two participating processes.

B - Incorrect: Compare-and-swap is an atomic checking/updating primitive, rather than the requested software semaphore abstraction.

C - Incorrect: Classic Peterson's algorithm solves the two-process case, not the stated more-than-two configuration.

D - Incorrect: Test-and-set is an atomic primitive; the software multiple-permit coordination mechanism here is a semaphore.

Primary note proof

Study-note section OS11

> A semaphore provides atomic wait/P/down and signal/V/up operations. Under the nonnegative-token model, wait consumes one available token or blocks when none is available; signal supplies a token and may wake a waiter. Implementations using negative internal counts encode waiters differently, so state the model. A binary semaphore represents one permit; a counting semaphore tracks several. A mutex normally has ownership rules that a general semaphore lacks.

Study-note section OS12

> Test-and-set and compare-and-swap combine checking/updating atomically; ordinary check-then-write does not. Peterson's classic algorithm uses flags plus turn for two processes under its memory-model assumptions; a semaphore can coordinate more than two.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

> • Loads a program into memory.
> • Executes the program.
> • Handles program's execution.
> • Provides a mechanism for process synchronization.
> • Provides a mechanism for process communication.
> • Provides a mechanism for deadlock handling.

Question sources

Mock exam 4 · Question 35

Question 16 | Q00359

A process can be terminated due to:

A. killed by another process

B. normal exit

C. all of the mentioned

D. fatal error

Hint

Separate creation, image replacement and termination; classify termination by its cause.

Answer C: All listed causes can terminate a process.

OS07 distinguishes voluntary normal completion from involuntary faults or being killed. Each named cause is therefore a genuine termination cause, although the causes differ in who initiates them and whether completion is normal. The aggregate choice includes all three. Selecting only one named cause would omit other valid termination paths supported by the complete note.

Why each choice is right or wrong

A - Incorrect: Being killed can terminate a process, but this alone omits normal completion and fatal faults.

B - Incorrect: Normal exit is valid, but it is only one of the three supported termination causes.

C - Correct: Being killed, normal exit and fatal error all provide legitimate process-termination causes.

D - Incorrect: A fatal error can terminate execution, but the other two named causes are also valid.

Primary note proof

Study-note section OS07

> Normal completion and a program-detected error exit are voluntary; an unrecoverable fault or being killed is involuntary. exit() terminates normally through the runtime; abort() requests abnormal termination.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 16 | lines 29-39

> 1. New The process is being created.
> 2.
> Ready The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run.
> 3.
> Running Process instructions are being executed (i.e. The process that is currently being

executed).
4.
Waiting The process is waiting for some event to occur (such as the completion of an I/O operation).
5. Terminated The process has finished execution.

Operating System module | PDF page 19 | lines 20-25

Non-Running Processes that are not running are kept in queue, waiting for their turn to execute. Each entry in the queue is a pointer to a particular process. Queue is implemented by using linked list. Use of dispatcher is as follows. When a process is interrupted, that process is transferred in the waiting queue. If the process has completed or aborted, the process is discarded. In either case, the dispatcher then selects a process from the queue to execute.

Question sources

Computer Science model exam · Question 90

University of Gondar model exam · Question 90

Question 17 | Q00360

Which named interface directly requests protected operating-system services across the user/kernel boundary, rather than merely providing application-facing wrappers?

A. library

B. system calls

C. assembly instructions

D. API

Hint

Identify the abstraction layer and separate a protected service request from an application task.

Answer B: system calls

A system call requests a protected kernel service, such as process creation or file access. An application may call an API implemented by a library, but that wrapper is distinct from the direct user/kernel service boundary. Assembly can encode the machine instructions used to enter that boundary without being its general interface name. The requested abstraction is therefore system calls.

Why each choice is right or wrong

A - Incorrect: A library can wrap system calls but is not itself the direct kernel-service boundary.

B - Correct: System calls directly request protected kernel services.

C - Incorrect: Assembly instructions are a programming representation, not the named service-interface abstraction.

D - Incorrect: An API is the application-facing interface and may wrap the direct system-call request.

Primary note proof

Study-note section OS04

> A system call requests a protected kernel service, such as creating a process, reading a file or opening a socket. A user-facing API/library may wrap that request; not every library function enters the kernel. Distinguish the application API from the direct user/kernel boundary when describing access to OS services.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 53 | lines 16-19

> implement most of the functionalities of the operating system and do not requires kernel module's code access rights.
> System Utility - System Utility programs are responsible to do specialized, individual level tasks.

Operating System module | PDF page 45 | lines 28-33

> Application I/O Interface represents the structuring techniques and interfaces for the operating system to enable I/O devices to be treated in a standard, uniform way. The actual differences lies kernel level modules called device drivers which are custom tailored to corresponding devices but show one of the standard interfaces to applications. The purpose of the device-driver layer is to hide the differences among device controllers from the I/O subsystem of the kernel, such as the I/O system calls. Following are the characteristics of I/O

Question sources

Computer Science model exam · Question 91

Operating Systems practice questions · Question 4

University of Gondar model exam · Question 91

Question 18 | Q00361

Which module gives control of the CPU to the process selected by the short-term scheduler?

A. Interrupt

B. Dispatcher

C. Scheduler

D. none of the mentioned

Hint

Decide whether the process is eligible, running or waiting, then identify the responsible queue or scheduling role.

Answer B: Dispatcher

The module uses overlapping terminology: it calls the short-term scheduler dispatcher on PDF p20, and describes the dispatcher selecting execution on p19. Under the conventional selection-versus-handoff reading requested by the stem, B is the matching term; p21 explains the handoff state work.

Why each choice is right or wrong

A - Incorrect: Interrupts trigger handling, not the named CPU-handoff module (p45).

B - Correct: Dispatcher is the execution-transfer role described on p19, with state switching on p21.

C - Incorrect: The stem already assigns selection to the scheduler; note that p20 loosely equates it with dispatcher.

D - Incorrect: The module explicitly names a dispatcher, so none is not appropriate.

Primary note proof

Study-note section OS06

> Long-term scheduling admits jobs; medium-term scheduling suspends/swaps them; short-term scheduling selects ready work. The dispatcher performs the handoff/context restore after selection.

Study-note section OS05

> A context switch saves one execution context and restores another; useful application work is interrupted by this overhead.

Study-note section OS34

> Polling checks readiness repeatedly; interrupts request CPU attention; DMA transfers blocks after setup without CPU work for every byte.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 19 | lines 20-25

> Non-Running Processes that are not running are kept in queue, waiting for their turn to execute. Each entry in the queue is a pointer to a particular process. Queue is implemented by using linked list. Use of dispatcher is as follows. When a process is interrupted, that process is transferred in the waiting queue. If the process has completed or aborted, the process is discarded. In either case, the dispatcher then selects a process from the queue to execute.

Operating System module | PDF page 20 | lines 15-21

> It is also called CPU scheduler. Main objective is increasing system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. CPU scheduler selects process among the processes that are ready to execute and allocates CPU to one of them.

Short term scheduler also known as dispatcher, execute most frequently and makes the fine-grained decision of which process to execute next. Short term scheduler is faster than long term scheduler.

Operating System module | PDF page 21 | lines 41-50

A context switch is the mechanism to store and restore the state or context of a CPU in Process Control block so that a process execution can be resumed from the same point at a later time. Using this technique, a context switcher enables multiple processes to share a single CPU. Context switching is an essential part of a multitasking operating system features.
When the scheduler switches the CPU from executing one process to execute another, the context switcher saves the content of all processor registers for the process being removed from the CPU, in its process descriptor. The context of a process is represented in the process control block of a process. Context switch time is pure overhead. Context switching can significantly affect performance as modern computers have a lot of general

Question sources

Computer Science model exam · Question 92

University of Gondar model exam · Question 92

Question 19 | Q00362

Which of the following condition is required for deadlock to be possible?

A. mutual exclusion

B. all of the mentioned

C. a process may hold allocated resources while awaiting assignment of other resources

D. no resource can be forcibly removed from a process holding it

Hint

Trace the resource dependencies and distinguish necessary conditions from evidence of an actual deadlock.

Answer B: All named conditions are necessary conditions.

OS22 lists mutual exclusion, hold-and-wait, no resource preemption and circular wait as necessary in the standard reusable-resource model. The three substantive choices name the first three, so all of the mentioned is the comprehensive option. Circular wait is also required despite being omitted from the list. Having these policies does not prove a particular current allocation is already deadlocked.

Why each choice is right or wrong

A - Incorrect: Mutual exclusion is necessary, but selecting it alone omits the other two necessary conditions offered.

B - Correct: Each substantive option names a standard necessary deadlock condition; the aggregate includes all offered conditions.

C - Incorrect: Retaining resources while requesting more is hold-and-wait, one valid condition among the three offered.

D - Incorrect: No forced removal is no-preemption, which is necessary but does not exclude the other valid conditions.

Primary note proof

Study-note section OS22

> Mutual exclusion allows at most one holder of a nonshareable resource. Hold-and-wait means retaining a resource while requesting another. No resource preemption means release must be voluntary. Circular wait forms a closed chain of such waits.

Study-note section OS22

> All four are necessary in the standard reusable-resource model.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

> • Loads a program into memory.
> • Executes the program.
> • Handles program's execution.
> • Provides a mechanism for process synchronization.
> • Provides a mechanism for process communication.
> • Provides a mechanism for deadlock handling.

Question sources

Computer Science model exam · Question 93

University of Gondar model exam · Question 93

Question 20 | Q00363

Which scheduling algorithm allocates the CPU first to the process that requests the CPU first?

A. first-come, first-served scheduling

B. priority scheduling

C. shortest job scheduling

D. none of the mentioned

Hint

Follow arrival order, include idle periods and distinguish waiting from completion time.

Answer A: first-come, first-served scheduling

Jobs execute first come, first served on PDF p23. The first CPU requester is consequently the first served in the FCFS queue.

Why each choice is right or wrong

A - Correct: Directly matches the arrival-order rule (p23).

B - Incorrect: Uses highest priority (p24).

C - Incorrect: Uses execution-length information (p23).

D - Incorrect: An offered policy already matches the rule.

Primary note proof

Study-note section OS15

> FCFS runs ready bursts in arrival order and ordinarily does not preempt.

Study-note section OS16

> Non-preemptive SJF/SJN chooses the shortest next burst among currently ready jobs.

Study-note section OS17

> Priority scheduling chooses the highest-priority ready job.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 23 | lines 12-17

> 7.1 First Come First Serve (FCFS)
> • Jobs are executed on first come, first serve basis.
> • Easy to understand and implement.
> • Poor in performance as average wait time is high.
> Wait time of each process is following
> Process Wait Time: Service Time - Arrival Time

Operating System module | PDF page 24 | lines 10-15

> 7.3 Priority Based Scheduling
> • Each process is assigned a priority. Process with highest priority is to be executed first and so on.
> • Processes with same priority are executed on first come first serve basis.
> • Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Operating System module | PDF page 23 | lines 23-26

> 7.2 Shortest Job First (SJF)
> • Best approach to minimize waiting time.
> • Impossible to implement
> • Processer should know in advance how much time process will take.

Variant 1 | Alternative version 1

Which scheduling algorithm allocates the CPU first to the process that requests the CPU first?

A. Priority scheduling

B. Shortest job scheduling

C. First-come, first-served scheduling

D. None of above

Answer C: First-come, first-served scheduling

Apply the same study-note rule to this alternative source set; the choice contrasts below show the derivation.

Why each choice is right or wrong

A - Incorrect: Priority scheduling chooses assigned rank, not request order (PDF p24).

B - Incorrect: Shortest-job scheduling chooses duration, not request order (PDF p23).

C - Correct: FCFS serves jobs first come, first served (PDF p23). The variant moves the correct text from current A to variant C.

D - Incorrect: Variant C supplies the arrival-order policy, so none does not apply (PDF p23).

Primary note proof

Study-note section OS15

> FCFS runs ready bursts in arrival order and ordinarily does not preempt.

Study-note section OS16

> Non-preemptive SJF/SJN chooses the shortest next burst among currently ready jobs.

Study-note section OS17

> Priority scheduling chooses the highest-priority ready job.

Question sources

Computer Science model exam · Question 94

University of Gondar model exam · Question 94

Computer Science model exam 2 · Question 75

Question 21 | Q00607

Which directory implementation is used in most Operating System?

A. Single level directory structure

B. Two level directory structure

C. Tree directory structure

D. Acyclic directory structure

Hint

Follow namespace structure and distinguish names, underlying files and mounted filesystems.

Answer C: Tree directory structure, in the usual general-purpose OS sense.

OS37 states that modern general-purpose systems commonly present hierarchical namespaces and defines a tree as allowing nested directories and absolute/relative paths. That directly supports the conventional tree-directory choice. This is a general-purpose namespace convention, not a measured claim about every OS ever built; links can add graph structure without replacing the hierarchy as the basic user-visible organization.

Why each choice is right or wrong

A - Incorrect: A single namespace without nested directories does not match the common hierarchical organization described.

B - Incorrect: Separating user directories alone is less general than the nested hierarchy specified for modern general-purpose systems.

C - Correct: Nested directories and paths form the tree/hierarchical organization the note identifies as commonly presented.

D - Incorrect: Links can add graph sharing, but acyclic sharing is not the basic common directory organization identified.

Primary note proof

Study-note section OS37

> A single-level directory requires unique names in one namespace. Two-level directories separate users. A tree allows nested directories and absolute/relative paths; links can create shared graphs. Modern general-purpose systems commonly present hierarchical namespaces, but a feature name alone does not prove that every system excludes simpler arrangements.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 6 | lines 19-23

> A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions. Operating System does the following activities for file management.
> Keeps track of information, location, uses, status etc. The collective facilities are often known as file system.

Operating System module | PDF page 54 | lines 5-17

> Portable - Portability means software can works on different types of hardware in same way. Linux kernel and application programs support their installation on any kind of hardware platform.
> Open Source - Linux source code is freely available and it is community-based

development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
Multi-User - Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.
Multiprogramming - Linux is a multiprogramming system means multiple applications can run at same time.
Hierarchical File System - Linux provides a standard file structure in which system files/ user files are arranged.
Shell - Linux provides a special interpreter program which can be used to execute

Question sources

Computer Science model exam · 2015 · Question 76

Question 22 | Q00696

In the basic virtual-memory models, which technique divides the address space into fixed-size units mapped to equal-size physical frames?

A. Paging

B. Segmentation

C. A&B

D. All

Hint

Split the address into page and offset, then apply the mapping and access checks.

Answer A: Paging

Paging divides addresses into fixed-size pages and maps them to physical frames of the same size. Segmentation instead uses variable-length logical units such as code and stack. Thus the fixed-size condition selects paging alone. Both combined alternatives include segmentation and therefore fail that condition; fixed partitions elsewhere in an operating system are outside the page-versus-segment comparison.

Why each choice is right or wrong

A - Correct: Paging uses fixed-size pages and corresponding equal-size frames.

B - Incorrect: Segments are variable-length logical units, so segmentation does not satisfy the fixed-unit condition.

C - Incorrect: Including segmentation makes this combined choice fail the fixed-size requirement.

D - Incorrect: All includes the variable-size segmentation alternative and is therefore incorrect.

Primary note proof

Study-note section OS30

Pages and physical frames have equal fixed size P.

Study-note section OS31

Segmentation uses logical variable-length units such as code, stack and arrays.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 34 | lines 47-56

External fragmentation is avoided by using paging technique. Paging is a technique in which physical memory is broken into blocks of the same size called pages (size is power of 2, between 512 bytes and 8192 bytes). When a process is to be executed, it's corresponding pages are loaded into any available memory frames.
Logical address space of a process can be non-contiguous and a process is allocated physical memory whenever the free memory frame is available. Operating system keeps track of all free frames. Operating system needs n free frames to run a program of size n pages.
Address generated by CPU is divided into
• Page number (p) -- page number is used as an index into a page table which contains base address of each page in physical memory.

Operating System module | PDF page 35 | lines 7-12

Segmentation is a technique to break memory into logical pieces where each piece represents a group of related information. For example, data segments or code segment for each process, data segment for operating system and so on. Segmentation can be implemented using or without using paging.
Unlike paging, segment is having varying sizes and thus eliminates internal fragmentation. External fragmentation still exists but to lesser extent.

Question sources

Computer Science model exam 2 · Question 37

Question 23 | Q00731

When does a page fault occur?

A. The page is present in memory.

B. Deadlock occurs.

C. The page does not present in memory.

D. Buffering occurs.

Hint

Separate a permitted nonresident access from an illegal access and trace the required handling steps.

Answer C: The page does not present in memory.

A demand-paging access to a nonresident page causes the trap (PDF p38). PDF p39 distinguishes a valid but absent page, which is brought in, from an invalid address, which terminates the process. C presupposes an actual reference to that page.

Why each choice is right or wrong

A - Incorrect: Resident page access proceeds normally (p38).

B - Incorrect: Deadlock is not the page-residency test on p38.

C - Correct: An accessed valid page absent from RAM causes demand paging (pp38-39).

D - Incorrect: Buffering is not the fault-triggering residency condition.

Primary note proof

Study-note section OS32

> Virtual memory provides translated, protected address spaces and allows nonresident portions of a program to be backed by storage. It does not add physical RAM or remove all capacity constraints. Demand paging brings a page in when accessed. On a nonresident access: trap, validate mapping/permissions, find a frame or choose a victim, save a dirty victim if needed, obtain contents, update mapping and restart.

Study-note section OS32

> Not every fault reads a disk: some allocate zero-filled pages or establish shared mappings. An illegal access is rejected, not automatically loaded.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 38 | lines 11-18

> Hardware support is required to distinguish between those pages that are in memory and those pages that are on the disk using the valid-invalid bit scheme. Where valid and invalid pages can be checked by checking the bit. Marking a page will have no effect if the process never attempts to access the page. While the process executes and accesses pages that are memory resident, execution proceeds normally.
> Access to a page marked invalid causes a page-fault trap. This trap is the result of the operating system's failure to bring the desired page into memory. But page fault can be handled as following.

Operating System module | PDF page 39 | lines 5-11

> Check an internal table for this process, to determine whether the reference was a valid or it was an invalid memory access.
> Step 2
> If the reference was invalid, terminate the process. If it was valid, but page have not yet brought in, page in the latter.
> Step 3 Find a free frame.
> Step 4 Schedule a disk operation to read the desired page into the newly allocated frame.

Question sources

Computer Science model exam 2 · Question 72

Computer Science model exam 2 · Question 90

Question 24 | Q00732

Which of the following is a condition that causes deadlock?

A. Mutual exclusion

B. Hold and wait

C. Circular wait

D. No preemption

E. All of these

Hint

Trace the resource dependencies and distinguish necessary conditions from evidence of an actual deadlock.

Answer E: All four are standard necessary deadlock conditions.

OS22 lists mutual exclusion, hold-and-wait, no-preemption and circular wait as the four necessary conditions in the standard reusable-resource model. Each substantive choice therefore names a valid deadlock condition, making all of these the comprehensive answer. The wording causes must not be read as saying one condition alone is sufficient; actual allocation/request dependencies still require inspection.

Why each choice is right or wrong

A - Incorrect: Mutual exclusion is one necessary condition, but the other three named conditions also belong to the standard set.

B - Incorrect: Hold-and-wait is necessary, but selecting it alone leaves out the other valid conditions.

C - Incorrect: Circular wait is part of the condition set, but it is not the sole valid listed condition.

D - Incorrect: No-preemption is necessary in this model, alongside the other three substantive choices.

E - Correct: All four substantive choices are standard necessary conditions; together their names give the complete condition set.

Primary note proof

Study-note section OS22

> Mutual exclusion allows at most one holder of a nonshareable resource. Hold-and-wait means retaining a resource while requesting another. No resource preemption means release must be voluntary. Circular wait forms a closed chain of such waits.

Study-note section OS22

All four are necessary in the standard reusable-resource model. Their availability as general system policies does not prove a deadlock exists at this instant; inspect actual allocations/requests.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

- Loads a program into memory.
- Executes the program.
- Handles program's execution.
- Provides a mechanism for process synchronization.
- Provides a mechanism for process communication.
- Provides a mechanism for deadlock handling.

Question sources

Computer Science model exam 2 · Question 73

Question 25 | Q00733

What is an operating system?

A. System service provider to the application programs

B. Collection of programs that manages hardware resources

C. Interface between the hardware and application programs

D. All of above

Hint

Separate broad resource-management goals from individual services or applications.

Answer D: All of above

The OS manages hardware and mediates users/program execution (PDF p5), and provides services to programs (p10). Thus each of A-C is a valid aspect of the definition and D collects them.

Why each choice is right or wrong

A - Incorrect: Program services are explicitly provided (p10).

B - Incorrect: Hardware and resource management are explicit (p5).

C - Incorrect: The mediation/interface role is explicit (p5), including program execution.

D - Correct: All three descriptions agree with pp5 and 10.

Primary note proof

Study-note section OS01

An OS allocates CPU time, memory, storage, devices and communication resources, and provides services to programs. Convenience means making tasks easy through consistent interfaces, files and device abstractions. Efficiency means using resources effectively; it is not synonymous with the fastest response for every process. Application development tools use OS services but application development itself is not a core resource-management function.

Study-note section OS04

A system call requests a protected kernel service, such as creating a process, reading a file or opening a socket.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 5 | lines 6-13

An operating System (OS) is an intermediary between users and computer hardware. It provides users an environment in which a user can execute programs conveniently and efficiently. In technical terms, it is software which manages hardware. An operating System controls the allocation of resources and services such as memory, processors, devices and information.
1.1 Definition
An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.

Operating System module | PDF page 10 | lines 5-15

An Operating System provides services to both the users and to the programs.
• It provides programs, an environment to execute.
• It provides users, services to execute the programs in a convenient manner.
Operating Systems can provide the following common services:
✓ Program execution
✓ I/O operations
✓ File System manipulation
✓ Communication
✓ Error Detection
✓ Resource Allocation
✓ Protection

Question sources

Computer Science model exam 2 · Question 74

Computer Science model exam 2 · Question 36

Operating Systems practice questions · Question 1

Question 26 | Q00735

What is the ready state of a process?

A. When process is unable to run until some task has been completed

B. When process is scheduled to run after some execution

C. When process is using the CPU

D. None of the mentioned

Hint

Decide whether the process is eligible, running or waiting, then identify the responsible queue or scheduling role.

Answer B: When process is scheduled to run after some execution

Ready means waiting to be assigned a processor (PDF p16). B is the nearest paraphrase, provided scheduled to run means eligible and waiting, not already executing.

Why each choice is right or wrong

A - Incorrect: Waiting for an event is the Waiting state.

B - Correct: Eligible to receive CPU next matches Ready under the stated reading.

C - Incorrect: Using the CPU defines Running.

D - Incorrect: Ready is represented by B under the stated interpretation.

Primary note proof

Study-note section OS06

> New processes are created/admitted; ready means eligible for CPU but not currently running. Running uses a CPU. Waiting/blocked means unable to proceed until an event; terminated means execution has ended. Running→ready follows time-slice expiry or another preemption. Running→waiting follows blocking I/O. I/O completion normally gives waiting→ready, not a guaranteed immediate running state.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 16 | lines 29-39

> 1. New The process is being created.
> 2.
> Ready The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run.
> 3.
> Running Process instructions are being executed (i.e. The process that is currently being executed).
> 4.
> Waiting The process is waiting for some event to occur (such as the completion of an I/O

operation).
5. Terminated The process has finished execution.

Question sources

Computer Science model exam 2 · Question 76

Question 27 | Q00736

In the standard reusable-resource model, a set of processes is deadlocked when _____.

A. Each process is terminated

B. All processes are trying to kill each other

C. Each process is blocked waiting for an event or resource that only another blocked member of the set can provide

D. None of the mentioned

Hint

Trace the resource dependencies and distinguish necessary conditions from evidence of an actual deadlock.

Answer C: Each process is blocked waiting for an event or resource that only another blocked member of the set can provide

Deadlock requires blocked participants whose needed event or resource can be supplied only by another blocked participant in the same set. The closed dependency prevents any member from progressing without intervention. Terminated processes are no longer waiting, and attempts to kill one another do not establish a resource dependency. The revised blocking option states the necessary relationship rather than merely predicting permanent delay.

Why each choice is right or wrong

A - Incorrect: Termination ends execution and does not define a set of blocked dependent processes.

B - Incorrect: Attempts to kill other processes do not establish deadlocked resource waits.

C - Correct: This states the closed dependency among blocked members that defines deadlock.

D - Incorrect: The revised C explicitly gives the deadlock definition, so none is false.

Primary note proof

Study-note section OS22

A set is deadlocked when each member waits for an event/resource that only another blocked member can provide. Mere waiting for another holder is normal blocking; indefinite unfair delay can be starvation without a cycle of dependencies.

Study-note section OS07

Normal completion and a program-detected error exit are voluntary; an unrecoverable fault or being killed is involuntary. exit() terminates normally through the runtime; abort() requests abnormal termination.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 16 | lines 29-39

1. New The process is being created.
2. Ready The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run.
3. Running Process instructions are being executed (i.e. The process that is currently being executed).
4. Waiting The process is waiting for some event to occur (such as the completion of an I/O operation).
5. Terminated The process has finished execution.

Operating System module | PDF page 10 | lines 23-28

- Loads a program into memory.
- Executes the program.
- Handles program's execution.
- Provides a mechanism for process synchronization.
- Provides a mechanism for process communication.
- Provides a mechanism for deadlock handling.

Question sources

Computer Science model exam 2 · Question 77

Question 28 | Q00737

In Operating Systems, which of the following is/are CPU scheduling algorithms?

A. Priority

B. Round Robin

C. Shortest Job First

D. All of the mentioned

Hint

Compare the ready jobs using the stated burst or remaining-time rule.

Answer D: All of the mentioned

PDF pp23-25 explicitly cover priority, RR and SJF in the scheduling chapter, so D collects three genuine CPU scheduling policies.

Why each choice is right or wrong

A - Incorrect: Priority ordering is defined on p24.

B - Incorrect: Quantum-based scheduling is defined on p25.

C - Incorrect: Duration-based SJF is defined on p23.

D - Correct: All three belong to the chapter's scheduling methods.

Primary note proof

Study-note section OS16

> Non-preemptive SJF/SJN chooses the shortest next burst among currently ready jobs.

Study-note section OS17

> Priority scheduling chooses the highest-priority ready job.

Study-note section OS18

> RR serves a ready queue cyclically for at most quantum q per turn.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 23 | lines 12-17

> 7.1 First Come First Serve (FCFS)
> • Jobs are executed on first come, first serve basis.
> • Easy to understand and implement.
> • Poor in performance as average wait time is high.
> Wait time of each process is following
> Process Wait Time: Service Time - Arrival Time

Operating System module | PDF page 23 | lines 23-26

> 7.2 Shortest Job First (SJF)
> • Best approach to minimize waiting time.
> • Impossible to implement
> • Processer should know in advance how much time process will take.

Operating System module | PDF page 24 | lines 10-15

> 7.3 Priority Based Scheduling
> • Each process is assigned a priority. Process with highest priority is to be executed first and so on.
> • Processes with same priority are executed on first come first serve basis.
> • Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Operating System module | PDF page 25 | lines 10-14

7.4 Round Robin Scheduling
• Each process is provided a fix time to execute called quantum.
• Once a process is executed for given time period. Process is preempted and other process executes for given time period.
• Context switching is used to save states of preempted processes.

Question sources

Computer Science model exam 2 · Question 78

Operating Systems practice questions · Question 3

Question 29 | Q00738

In which of the storage placement strategies a program is placed in the largest available hole in the main memory?

A. Best fit

B. First fit

C. Worst fit

D. Buddy

Hint

Apply the stated hole-selection or base-and-bound rule with its explicit convention.

Answer C: Worst-fit.

OS28 defines worst-fit as selecting the largest hole, directly matching the stem. Best-fit chooses the smallest adequate hole, and first-fit stops at the first adequate one rather than comparing all sizes. The buddy label does not state this largest-hole rule in the provided note. No numeric trace is necessary because the selection criterion itself is explicit.

Why each choice is right or wrong

A - Incorrect: Best-fit chooses the smallest adequate hole, which is the opposite size-selection criterion from the stem.

B - Incorrect: First-fit selects the first adequate hole in search order, not necessarily the largest available hole.

C - Correct: Worst-fit is explicitly the policy that selects the largest available adequate hole.

D - Incorrect: The supplied largest-hole criterion is defined as worst-fit; the buddy label is not that rule.

Primary note proof

Study-note section OS28

First-fit selects the first adequate hole; next-fit continues from the previous search position; best-fit selects the smallest adequate hole; worst-fit selects the largest.

Study-note section OS-S03

A buddy allocator manages power-of-two block sizes. It takes a smallest sufficient block, splitting larger blocks when necessary, and can merge a freed block with its free buddy. This is not the worst-fit rule of selecting the largest arbitrary hole.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 34 | lines 34-45

External
fragmentation
Total memory space is enough to satisfy a request or to reside a
process in it, but it is not contiguous so it cannot be used.
2
Internal
fragmentation
Memory block assigned to process is bigger. Some portion of memory
is left unused as it cannot be used by another process.
External fragmentation can be reduced by compaction or shuffle memory contents to place all free memory together in one large block. To make compaction feasible, relocation should be dynamic

Question sources

Computer Science model exam 2 · Question 79

Question 30 | Q00916

Which one of the following function of operating system is categorized under process management?

A. Allocates the device in the efficient way

B. De-allocates the resources.

C. .
Allocates the memory when the process requests it to do so.

D. Allocates the processor (CPU) to a process

Hint

Separate broad resource-management goals from individual services or applications.

Answer D: Allocates the processor (CPU) to a process

CPU allocation is explicitly process/processor management (PDF p6). Memory requests belong to memory management (p5), and device allocation belongs to device management (p6).

Why each choice is right or wrong

A - Incorrect: Device allocation is listed under device management.

B - Incorrect: Resource deallocation is too broad and appears in several management categories, not this precise processor task.

C - Incorrect: Memory allocation is explicitly under memory management (p5).

D - Correct: Allocating CPU to a process is explicitly listed under processor management (p6).

Primary note proof

Study-note section OS01

> An OS allocates CPU time, memory, storage, devices and communication resources, and provides services to programs.

Study-note section OS06

> Long-term scheduling admits jobs; medium-term scheduling suspends/swaps them; short-term scheduling selects ready work.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 6 | lines 4-9

> In multiprogramming environment, OS decides which process gets the processor when and how much time. This function is called process scheduling. Operating System does the following activities for processor management. Keeps tracks of processor and status of process. Program responsible for this task is known as traffic controller.
> Allocates the processor (CPU) to a process.
> De-allocates processor when processor is no longer required.

Operating System module | PDF page 5 | lines 26-36

> Memory management refers to management of Primary Memory or Main Memory. Main memory is a large array of words or bytes where each word or byte has its own address. Main memory provides a fast storage that can be access directly by the CPU. So, for a program to be executed, must in the main memory.
> Activities Operating System for memory management.
> Keeps tracks of primary memory (i.e. what part of it are in use by whom, what part are not in use.)
> In multiprogramming, OS decides which process will get memory when and how much.
> Allocates the memory when the process requests it to do so.
> De-allocates the memory when the process no longer needs it or has been terminated.

Operating System module | PDF page 6 | lines 11-17

> OS manages device communication via their respective drivers. Activities of Operating System for device management.
> Keeps tracks of all devices. Program responsible for this task is known as the I/O

controller.
Decides which process gets the device when and for how much time.
Allocates the device in the efficient way.
De-allocates devices.

Question sources

Computer Science Exit Exam · 2018 · Question 29

Question 31 | Q00928

What is the primary function of an operating system?

A. To connect to the internet

B. To create documents and spreadsheets

C. .
To design computer hardware

D. To act as an intermediary between users and computer hardware

Hint

Separate broad resource-management goals from individual services or applications.

Answer D: To act as an intermediary between users and computer hardware

PDF p5 defines the OS as intermediary between users and hardware and controller of program execution. D states this general role; the other options are particular applications or other activities.

Why each choice is right or wrong

A - Incorrect: Internet connection is narrower than the whole OS definition.

B - Incorrect: Document authoring is a program activity, not the definition on p5.

C - Incorrect: Hardware design is not managing hardware during execution.

D - Correct: Directly matches the intermediary definition on p5.

Primary note proof

Study-note section OS01

An OS allocates CPU time, memory, storage, devices and communication resources, and provides services to programs. Convenience means making tasks easy through consistent interfaces, files and device abstractions. Efficiency means using resources effectively; it is not synonymous with the fastest response for every process. Application development tools use OS services but application development itself is not a core resource-management function.

Study-note section OS03

A shell reads/interprets user commands; a GUI is another interface. Browsers, office tools and editors are applications.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 5 | lines 6-13

An operating System (OS) is an intermediary between users and computer hardware. It provides users an environment in which a user can execute programs conveniently and efficiently. In technical terms, it is software which manages hardware. An operating System controls the allocation of resources and services such as memory, processors, devices and information.
1.1 Definition
An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.

Question sources

Computer Science Exit Exam · 2018 · Question 41

Question 32 | Q00935

A user installs a new printer. The OS automatically detects and configures it. This is an example of:

A. Plug-and-play functionality

B. Manual driver installation

C. .
Manual hardware management

D. Resource allocation

Hint

Separate device hardware, device-specific software and general event or transfer mechanisms.

Answer A: Plug-and-play functionality.

OS34 defines plug-and-play as discovering and configuring compatible new hardware, with the qualification that a driver may still be required. That is exactly the printer scenario. The automatic wording excludes the two manual alternatives. Resource allocation may occur during configuration, but it is broader than the specific automatic detection/configuration feature requested by the question.

Why each choice is right or wrong

A - Correct: Automatically discovering and configuring the installed printer is precisely the plug-and-play behavior described.

B - Incorrect: The scenario explicitly says automatic detection/configuration rather than manual driver installation by the user.

C - Incorrect: Manual hardware management conflicts with the automatic configuration behavior specified in the scenario.

D - Incorrect: Resources may be assigned during setup, but the specific automatic discovery feature is plug-and-play.

Primary note proof

Study-note section OS34

> Plug-and-play discovers/configures compatible new hardware but may still require a driver.

Study-note section OS34

> A driver converts OS requests into device-specific operations. Device independence and uniform naming hide brand-specific details; error handling reports/retries supported failures.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 30-37

> I/O subsystem comprised of I/O devices and their corresponding driver software. Drivers hides the peculiarities of specific hardware devices from the user as the device driver knows the peculiarities of the specific device.
> Operating System manages the communication between user and device drivers.
> The major activities of an operating system with respect to I/O Operation.
> • I/O operation means read or write operation with any file or any specific I/O device.
> • Program may require any I/O device while running.
> • Operating system provides the access to the required I/O device when required.

Operating System module | PDF page 47 | lines 19-26

> 12.5 Device driver
> Device driver is a program or routine developed for an I/O device. A device driver implements I/O operations or behaviors on a specific class of devices. For example a system supports one or a number of multiple brands of terminals, all slightly different terminals may have a single terminal driver. In the layered structure of I/O system, device driver lies between interrupt handler and device independent I/O software. The job of a device driver is following.
> To accept request from the device independent software above it.
> To see to it that the request is executed.

Question sources

Computer Science Exit Exam · 2018 · Question 48

Question 33 | Q00944

In asystem, resources cannot be forcibly taken from processes; they must be released voluntarily. Which deadlock condition does this represent?

A. No Preemption

B. Hold and Wait

C. .
Mutual Exclusion

D. Circular Wait

Hint

Trace the resource dependencies and distinguish necessary conditions from evidence of an actual deadlock.

Answer A: No preemption.

OS22 directly defines no resource preemption as requiring voluntary release. The stem repeats that resources cannot be forcibly taken, so the match is exact. Hold-and-wait concerns retaining one resource while requesting another, mutual exclusion concerns exclusive possession, and circular wait concerns a closed dependency chain. None of those separately states the prohibition on forced withdrawal.

Why each choice is right or wrong

A - Correct: Prohibiting forced withdrawal and requiring voluntary release is exactly the no-preemption resource condition.

B - Incorrect: Hold-and-wait concerns retaining a resource while requesting another, not whether forced removal is permitted.

C - Incorrect: Mutual exclusion limits concurrent holders; it does not itself require voluntary release.

D - Incorrect: Circular wait describes a closed chain of dependencies rather than the resource-withdrawal rule.

Primary note proof

Study-note section OS22

> Mutual exclusion allows at most one holder of a nonshareable resource. Hold-and-wait means retaining a resource while requesting another. No resource preemption means release must be voluntary. Circular wait forms a closed chain of such waits.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

> • Loads a program into memory.
> • Executes the program.

- Handles program's execution.
- Provides a mechanism for process synchronization.
- Provides a mechanism for process communication.
- Provides a mechanism for deadlock handling.

Question sources

Computer Science Exit Exam · 2018 · Question 57

Question 34 | Q00984

How does an operating system improve efficiency in a multi-user environment?

A. By allowing only one user to access the system at a time

B. By disabling all background services

C. .
By using time-sharing and process scheduling to allocate CPU time fairly among users and processes

D. By requiring manual intervention for each task

Hint

Compare interaction, CPU sharing and deadline requirements under the stated system model.

Answer C: .
By using time-sharing and process scheduling to allocate CPU time fairly among users and processes

Time-sharing switches CPU among users and gives each a small portion of time (PDF p7), reducing idle time and giving quick response. C captures that scheduling-based allocation.

Why each choice is right or wrong

A - Incorrect: One user at a time contradicts the simultaneous time-sharing service on p7.

B - Incorrect: Disabling background services is not the mechanism described.

C - Correct: CPU scheduling and time-sharing are explicitly the mechanism on p7.

D - Incorrect: Manual intervention per task is not the CPU-switching mechanism.

Primary note proof

Study-note section OS02

Batch jobs run with little interaction after submission. Multiprogramming keeps several jobs available so another can use the CPU while one waits; it does not require fixed time slices. Time-sharing uses rapid scheduling/preemption to support interactive users. A single CPU interleaves execution; multiple CPUs can execute work truly in parallel. Shared-memory multiprocessors do not require each CPU to have private main memory.

Study-note section OS20

Fair-share scheduling allocates according to stated weights, potentially per user/group rather than per process.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 7 | lines 20-33

Time sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time. Time-sharing or multitasking is a logical extension of multiprogramming. Processor's time which is shared among multiple users simultaneously is termed as time-sharing. The main difference between Multi-programmed Batch Systems and Time-Sharing Systems is that in case of multi-programmed batch systems, objective is to maximize processor use, whereas in Time-Sharing Systems objective is to minimize response time.
Multiple jobs are executed by the CPU by switching between them, but the switches occur so frequently. Thus, the user can receive an immediate response. For example, in a transaction processing, processor execute each user program in a short burst or quantum of computation. That is if n users are present, each user can get time quantum. When the user submits the command, the response time is in few seconds at most.
Operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time. Computer systems that were designed primarily as batch systems

Question sources

Computer Science Exit Exam · 2018 · Question 97

Question 35 | Q00994

Which exits are involuntary? Here “Terminate exit” means an externally imposed kill, “Fatal exit” means an unrecoverable execution fault, and “Error exit” means the program chooses to exit after detecting bad input. Select all that apply.

A. Normal exit

B. Terminate exit

C. Fatal exit

D. Error exit

Hint

Separate creation, image replacement and termination; classify termination by its cause.

Answer B, C: Terminate exit; Fatal exit

An externally imposed kill and an unrecoverable execution fault stop a process involuntarily. Normal completion is chosen by the program, and detecting an input error then exiting is also a voluntary program decision. The meanings separate causes of termination instead of treating every use of the word terminate as one category. Thus the external-kill and fatal-fault choices are selected.

Why each choice is right or wrong

A - Incorrect: Normal completion is a voluntary exit.

B - Correct: The defined external kill imposes termination and is involuntary.

C - Correct: An unrecoverable execution fault imposes termination and is involuntary.

D - Incorrect: The program-detected error exit is a voluntary decision under the stated meaning.

Primary note proof

Study-note section OS07

> Normal completion and a program-detected error exit are voluntary; an unrecoverable fault or being killed is involuntary. exit() terminates normally through the runtime; abort() requests abnormal termination.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 16 | lines 29-39

> 1. New The process is being created.
> 2.
> Ready The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run.
> 3.
> Running Process instructions are being executed (i.e. The process that is currently being executed).
> 4.
> Waiting The process is waiting for some event to occur (such as the completion of an I/O operation).
> 5. Terminated The process has finished execution.

Operating System module | PDF page 39 | lines 5-11

> Check an internal table for this process, to determine whether the reference was a valid or it was an invalid memory access.
> Step 2
> If the reference was invalid, terminate the process. If it was valid, but page have not yet brought in, page in the latter.
> Step 3 Find a free frame.
> Step 4 Schedule a disk operation to read the desired page into the newly allocated frame.

Question sources

National Exit Exam practice questions · Question 8

Computer Science Exit Exam practice collection · Question 217

Question 36 | Q01085

How do we define the term Thread?

A. Device that controls input

B. Variable that controls movement

C. Controlled execution of applications

D. None of the above

Hint

Compare per-thread execution context with shared process resources and the stated thread mapping.

Answer D: None of the above

A thread is a separate flow of execution through process code with its own PC, registers and stack (PDF p27). None of A-C states that definition; controlled execution of applications is too broad to identify a thread.

Why each choice is right or wrong

A - Incorrect: A device is not the defined flow of execution.

B - Incorrect: A variable is not a flow of control with a stack and PC.

C - Incorrect: Generic controlled application execution does not distinguish thread from process or scheduler.

D - Correct: None of A-C gives the thread definition on p27.

Primary note proof

Study-note section OS08

> A thread has its own PC, registers and stack. Threads in a process share code, address space and usually open-file resources. A process can contain many threads; a thread belongs to one process at a time. Threads are not a group of processes and need not execute sequentially. Independent processes can exist; cooperating ones communicate/share data.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 27 | lines 6-14

> A thread is a flow of execution through the process code, with its own program counter, system registers and stack. A thread is also called a light weight process. Threads provide a

way to improve application performance through parallelism. Threads represent a software approach to improving performance of operating system by reducing the overhead thread is equivalent to a classical process.
Each thread belongs to exactly one process and no thread can exist outside a process. Each thread represents a separate flow of control. Threads have been successfully used in implementing network servers and web server. They also provide a suitable foundation for parallel execution of applications on shared memory multiprocessors. Following figure shows

Question sources

Web Programming practice questions · Question 18

Question 37 | Q01186

Where are the list of processes that are prepared to be executed and waiting placed?

A. Job queue

B. Ready queue

C. Execution queue

D. Process queue

Hint

Decide whether the process is eligible, running or waiting, then identify the responsible queue or scheduling role.

Answer B: Ready queue

PDF p19 says processes wait in the ready queue for CPU allocation. That is exactly prepared to execute but awaiting a processor.

Why each choice is right or wrong

A - Incorrect: Job queue contains all processes in the system (p18), rather than specifically ready ones.

B - Correct: Ready queue is the explicit CPU-waiting queue (p19).

C - Incorrect: No separate execution queue is defined for this purpose.

D - Incorrect: Process queue is generic and not the specific named queue on p19.

Primary note proof

Study-note section OS06

The ready queue contains eligible work; device queues hold blocked requests. The broad textbook job queue contains processes in the system, while an admission queue more narrowly holds jobs awaiting memory/admission. Long-term scheduling admits jobs; medium-term scheduling suspends/swaps them; short-term scheduling selects ready work.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 19 | lines 6-12

> A newly arrived process is put in the ready queue. Processes waits in ready queue for allocating the CPU. Once the CPU is assigned to a process, then that process will execute. While executing the process, any one of the following events can occur.
> • The process could issue an I/O request and then it would be placed in an I/O queue.
> • The process could create new sub process and will wait for its termination.
> • The process could be removed forcibly from the CPU, as a result of interrupt and put back in the ready queue.

Operating System module | PDF page 18 | lines 13-18

> The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.
> Process scheduling is an essential part of a Multiprogramming operating system. Such operating systems allow more than one process to be loaded into the executable memory at a time and loaded process shares the CPU using time multiplexing.

Question sources

Computer Science model exam 2 · Question 7

Question 38 | Q01192

CPU scheduling is the basis of _____

A. Multiprogramming operating systems

B. Larger memory-sized systems

C. Multiprocessor systems

D. None of the mentioned

Hint

Compare interaction, CPU sharing and deadline requirements under the stated system model.

Answer A: Multiprogramming operating systems

Process scheduling is explicitly essential to a multiprogramming OS (PDF p18), enabling loaded processes to time-share one CPU; multiple processors or unusually large RAM are not required.

Why each choice is right or wrong

A - Correct: Multiprogramming explicitly needs this CPU sharing (pp13,18).

B - Incorrect: Large RAM is not the defining scheduling basis.

C - Incorrect: Multiprogramming can assume a single shared processor (p13).

D - Incorrect: The module explicitly names multiprogramming.

Primary note proof

Study-note section OS02

> Multiprogramming keeps several jobs available so another can use the CPU while one waits; it does not require fixed time slices. Time-sharing uses rapid scheduling/preemption to support interactive users. A single CPU interleaves execution; multiple CPUs can execute work truly in parallel.

Study-note section OS06

> Long-term scheduling admits jobs; medium-term scheduling suspends/swaps them; short-term scheduling selects ready work.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 18 | lines 13-18

> The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.
> Process scheduling is an essential part of a Multiprogramming operating system. Such operating systems allow more than one process to be loaded into the executable memory at a time and loaded process shares the CPU using time multiplexing.

Operating System module | PDF page 13 | lines 21-25

> 4.3 Multiprogramming
> When two or more programs are residing in memory at the same time, then sharing the processor is referred to the multiprogramming. Multiprogramming assumes a single shared processor. Multiprogramming increases CPU utilization by organizing jobs so that the CPU always has one to execute.

Question sources

Computer Science model exam 2 · Question 13

Operating Systems practice questions · Question 5

Question 39 | Q01251

An OS designer must choose between checking for deadlock after every resource request and checking periodically. No detection cost, request rate or acceptable delay is specified. Which conclusion about a universally best checking schedule is justified?

A. Both checking on every request and checking periodically are always equally optimal

B. Checking at fixed time intervals is always optimal

C. Checking on every resource request is always optimal

D. Neither schedule is universally best; the choice trades detection overhead against delay

Hint

Separate handling before allocation from actions taken after detection, and consider the cost of checking.

Answer D: Neither schedule is universally best; the choice trades detection overhead against delay

Checking after every request can discover a problem promptly but incurs repeated detection overhead. Periodic checking reduces check frequency while allowing a deadlock to persist until the next check. Which tradeoff is suitable depends on workload and acceptable detection delay. Without those criteria neither schedule is universally optimal, nor is there a basis for claiming they are always equally good.

Why each choice is right or wrong

A - Incorrect: The two policies need not have equal overhead or detection delay.

B - Incorrect: Fixed intervals may suit some workloads, but no universal optimum follows.

C - Incorrect: Per-request checks can impose excessive overhead despite earlier detection.

D - Correct: This states the overhead-versus-delay tradeoff without asserting an unsupported universal optimum.

Primary note proof

Study-note section OS26

> Checking on every request versus periodically trades overhead against detection delay; no universal best interval exists.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

> • Loads a program into memory.
> • Executes the program.
> • Handles program's execution.
> • Provides a mechanism for process synchronization.
> • Provides a mechanism for process communication.
> • Provides a mechanism for deadlock handling.

Question sources

Computer Science model exam 2 · Question 72

Question 40 | Q01265

Which listed operation is directly an OS/tool task? Here “bad-block recovery” specifically means autonomous drive-internal sector remapping, and “booting from disk” means the initial power-on stage before the kernel has been loaded.

A. bad-block recovery

B. booting from disk

C. disk initialization

D. all of the mentioned

Hint

Identify whether data is in transit, retained for reuse or queued for a device.

Answer C: disk initialization

Disk initialization and filesystem creation are identified as OS/tool work. Firmware starts the boot sequence before the kernel takes control, while a drive may use its own firmware to remap defective sectors. The question asks about that direct responsibility boundary, rather than broad coordination of every disk-related activity. Consequently disk initialization is the only matching choice, and the aggregate is too broad.

Why each choice is right or wrong

A - Incorrect: The specified bad-sector remapping is drive-firmware work, not the requested OS/tool operation.

B - Incorrect: The stage beginning boot is firmware work before kernel control.

C - Correct: Disk initialization/filesystem preparation is OS/tool work.

D - Incorrect: The other operations include firmware responsibilities, so all is too broad.

Primary note proof

Study-note section OS35

> Disk initialization/filesystem creation is OS/tool work, firmware begins boot, and drive firmware may remap bad sectors.

Study-note section OS03

> Firmware/bootloader stages execute before transferring control to a kernel loaded into RAM.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 6 | lines 11-17

> OS manages device communication via their respective drivers. Activities of Operating System for device management.
> Keeps tracks of all devices. Program responsible for this task is known as the I/O controller.

Decides which process gets the device when and for how much time.
Allocates the device in the efficient way.
De-allocates devices.

Operating System module | PDF page 47 | lines 17-18

Error Handling - An operating system that uses protected memory can guard against many kinds of hardware and application errors.

Question sources

Computer Science model exam 2 · Question 86

Question 41 | Q01274

Which method is used to prevent threads or processes from accessing a single resource?

A. PCB

B. Semaphore

C. Job Scheduler

D. Non-Contiguous Memory Allocation

Hint

Track available permits and blocked operations using the stated semaphore model.

Answer B: A semaphore can enforce one-at-a-time access.

OS11 describes a binary semaphore as one permit and gives wait(m); critical work; signal(m) with m initialized to 1. This prevents competing threads/processes from entering the protected resource operation simultaneously; it does not mean nobody can ever use the resource. A PCB stores process state, a scheduler selects jobs, and allocation layout does not enforce this resource-access rule.

Why each choice is right or wrong

A - Incorrect: A PCB records execution and resource state rather than supplying atomic mutual-exclusion operations.

B - Correct: A one-permit semaphore with atomic wait/signal can serialize entry to the protected resource operation.

C - Incorrect: A job scheduler selects admitted or ready work; it does not by itself protect shared resource accesses.

D - Incorrect: Noncontiguous memory placement changes allocation layout rather than synchronizing access to one shared resource.

Primary note proof

Study-note section OS11

A semaphore provides atomic wait/P/down and signal/V/up operations. Under the nonnegative-token model, wait consumes one available token or blocks when none is available; signal supplies a token and may wake a waiter. Implementations using negative internal counts encode waiters differently, so state the model. A binary semaphore represents one permit; a counting semaphore tracks several. A mutex normally has ownership rules that a general semaphore lacks.

Study-note section OS11

Initialize a mutex semaphore to 1. Each thread does wait(m); critical work; signal(m). For a buffer of N slots use empty=N, full=0 and mutex=1: producer waits empty then mutex, inserts, unlocks then signals full. Waiting for empty while already holding mutex can prevent the consumer from freeing a slot.

Study-note section OS05

A saved PC identifies where execution resumes.

Study-note section OS06

Long-term scheduling admits jobs; medium-term scheduling suspends/swaps them; short-term scheduling selects ready work.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

• Loads a program into memory.
• Executes the program.
• Handles program's execution.
• Provides a mechanism for process synchronization.
• Provides a mechanism for process communication.
• Provides a mechanism for deadlock handling.

Operating System module | PDF page 17 | lines 4-7

Each process is represented in the operating system by a process control block (PCB) also called a task control block. PCB is the data structure used by the operating system. Operating system groups all information that needs about particular process. PCB contains many pieces of information associated with a specific process which is described below.

Operating System module | PDF page 20 | lines 15-21

It is also called CPU scheduler. Main objective is increasing system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. CPU scheduler selects process among the processes that are ready to execute and allocates CPU to one of them.
Short term scheduler also known as dispatcher, execute most frequently and makes the

fine-grained decision of which process to execute next. Short term scheduler is faster than long term scheduler.

Operating System module | PDF page 34 | lines 47-56

External fragmentation is avoided by using paging technique. Paging is a technique in which physical memory is broken into blocks of the same size called pages (size is power of 2, between 512 bytes and 8192 bytes). When a process is to be executed, it's corresponding pages are loaded into any available memory frames.
Logical address space of a process can be non-contiguous and a process is allocated physical memory whenever the free memory frame is available. Operating system keeps track of all free frames. Operating system needs n free frames to run a program of size n pages.
Address generated by CPU is divided into
• Page number (p) -- page number is used as an index into a page table which contains base address of each page in physical memory.

Question sources

Computer Science model exam 2 · Question 95

Question 42 | Q01283

Which of the following is not a type of operating system?

A. Windows

B. Linux

C. macOS

D. Microsoft Office

Hint

Use the stated historical or platform scope and distinguish origin from technical capability.

Answer D: Microsoft Office is an application suite.

OS40 explicitly identifies Microsoft Office as an application suite and discusses Windows, Linux and macOS as OS/kernel families. Office runs using OS services rather than providing the core resource-management platform. Thus Office is the exclusion in this list. The kernel-versus-distribution distinction for Linux does not turn it into an office application or invalidate the conventional OS-family contrast.

Why each choice is right or wrong

A - Incorrect: Windows is identified as the Microsoft operating-system family, so it is not the application-suite exclusion.

B - Incorrect: Linux names a kernel used with distributions as operating systems, unlike the Office application suite.

C - Incorrect: macOS is an operating-system product rather than a user application suite in this classification.

D - Correct: Microsoft Office is explicitly described as an application suite, not an operating system.

Primary note proof

Study-note section OS40

> Linux is a Unix-compatible kernel written from scratch by Linus Torvalds; a distribution combines it with userland, libraries and tools. Android's base kernel layer is Linux; its framework and applications are higher layers. AOSP is open source, while a shipped device can include proprietary applications/services. Windows and macOS are both proprietary products, although macOS contains open-source components, so "which is not open source?" offering both is ambiguous. Microsoft Office is an application suite.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 8 | lines 21-26

> Network Operating System runs on a server and provides server the capability to manage data, users, groups, security, applications, and other networking functions. The primary purpose of the network operating system is to allow shared file and printer access among multiple computers in a network, typically a local area network (LAN), a private network or to other networks. Examples of network operating systems are Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD.

Operating System module | PDF page 53 | lines 5-7

> Linux is one of popular version of UNIX operating System. It is open source as its source code is freely available. It is free to use. Linux was designed considering UNIX compatibility. Its functionality list is quite similar to that of UNIX.

Question sources

Operating Systems practice questions · Question 1

Question 43 | Q01284

What is the main function of an operating system?

A. To manage hardware resources

B. To manage software applications

C. To manage user accounts

D. To manage network connections

Hint

Separate broad resource-management goals from individual services or applications.

Answer A: To manage hardware resources

Hardware resource management is the general technical definition on PDF p5. Applications, accounts and connections are narrower services and cannot replace that umbrella definition.

Why each choice is right or wrong

A - Correct: Matches the hardware-management definition (p5).

B - Incorrect: Application execution is a service within that broader role.

C - Incorrect: Accounts are one aspect of network/user administration (p8).

D - Incorrect: Network connection management is narrower than OS-wide hardware management (p8).

Primary note proof

Study-note section OS01

> An OS allocates CPU time, memory, storage, devices and communication resources, and provides services to programs. Convenience means making tasks easy through consistent interfaces, files and device abstractions. Efficiency means using resources effectively; it is not synonymous with the fastest response for every process. Application development tools use OS services but application development itself is not a core resource-management function.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 5 | lines 6-13

> An operating System (OS) is an intermediary between users and computer hardware. It provides users an environment in which a user can execute programs conveniently and efficiently. In technical terms, it is software which manages hardware. An operating System controls the allocation of resources and services such as memory, processors, devices and information.
> 1.1 Definition
> An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.

Operating System module | PDF page 8 | lines 21-26

> Network Operating System runs on a server and provides server the capability to manage data, users, groups, security, applications, and other networking functions. The primary purpose of the network operating system is to allow shared file and printer access among multiple computers in a network, typically a local area network (LAN), a private network or to other networks. Examples of network operating systems are Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD.

Question sources

Operating Systems practice questions · Question 2

Question 44 | Q01285

Which of the following is not a memory management technique used in operating systems?

A. Paging

B. Segmentation

C. Swapping

D. Virtualization

Hint

Identify the abstraction layer and separate a protected service request from an application task.

Answer D: Virtualization, in the virtual-machine sense.

OS04 defines virtualization as presenting virtual machine environments and distinguishes it from virtual memory. OS27 describes swapping, OS30 paging and OS31 segmentation as memory-management mechanisms. Under the guest-machine sense used here, virtualization is therefore the option outside that list. The answer depends on that stated sense; the broader phrase memory virtualization would describe a different use of the word.

Why each choice is right or wrong

A - Incorrect: Paging maps fixed-size virtual pages to physical frames and is explicitly a memory-management mechanism.

B - Incorrect: Segmentation maps variable-length logical regions and is explicitly a memory-management mechanism.

C - Incorrect: Swapping moves a process to backing store and is a memory-management operation described here.

D - Correct: In the note's guest-machine sense, virtualization presents virtual computers rather than naming these process-memory techniques.

Primary note proof

Study-note section OS04

> Virtualization presents a virtual machine environment, often allowing guest operating systems on one host through a hypervisor. It differs from virtual memory, which translates a process's addresses. Kernel, shell, API and VM are not interchangeable labels.

Study-note section OS27

> Swapping traditionally moves a whole process to backing store; paging moves individual pages.

Study-note section OS30

> Pages and physical frames have equal fixed size P. Divide virtual address v into page=floor(v/P) and offset=v mod P. The page table maps a page to a frame and protection/status flags.

Study-note section OS31

Segmentation uses logical variable-length units such as code, stack and arrays.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 34 | lines 47-56

External fragmentation is avoided by using paging technique. Paging is a technique in which physical memory is broken into blocks of the same size called pages (size is power of 2, between 512 bytes and 8192 bytes). When a process is to be executed, it's corresponding pages are loaded into any available memory frames.
Logical address space of a process can be non-contiguous and a process is allocated physical memory whenever the free memory frame is available. Operating system keeps track of all free frames. Operating system needs n free frames to run a program of size n pages.
Address generated by CPU is divided into
• Page number (p) -- page number is used as an index into a page table which contains base address of each page in physical memory.

Operating System module | PDF page 35 | lines 7-12

Segmentation is a technique to break memory into logical pieces where each piece represents a group of related information. For example, data segments or code segment for each process, data segment for operating system and so on. Segmentation can be implemented using or without using paging.
Unlike paging, segment is having varying sizes and thus eliminates internal fragmentation. External fragmentation still exists but to lesser extent.

Operating System module | PDF page 33 | lines 15-19

Swapping is a mechanism in which a process can be swapped temporarily out of main memory to a backing store, and then brought back into memory for continued execution.
Backing store is a usually a hard disk drive or any other secondary storage which fast in access and large enough to accommodate copies of all memory images for all users. It must be capable of providing direct access to these memory images.

Operating System module | PDF page 37 | lines 6-11

Virtual memory is a technique that allows the execution of processes which are not completely available in memory. The main visible advantage of this scheme is that programs can be larger than physical memory. Virtual memory is the separation of user logical memory from physical memory.
This separation allows an extremely large virtual memory to be provided for programmers when only a smaller physical memory is available. Following are the situations, when entire

Question sources

Operating Systems practice questions · Question 3

Question 45 | Q01286

Which of the following is not a process scheduling algorithm?

A. First-Come-First-Serve (FCFS)

B. Round-Robin

C. Shortest-Job-First (SJF)

D. Binary Search Tree (BST)

Hint

Separate class assignment, movement between queues and the rule for choosing work.

Answer D: Binary Search Tree is a data structure.

OS19 explicitly states that a binary search tree may implement a queue but is not itself a CPU selection policy. FCFS, Round Robin and SJF are actual scheduling algorithms described in OS15, OS18 and OS16. Consequently BST is the exclusion. Using a tree internally to maintain priorities would not change the selection policy into an algorithm called BST.

Why each choice is right or wrong

A - Incorrect: FCFS is a scheduling policy that runs ready bursts in their arrival order.

B - Incorrect: Round Robin is a scheduling policy assigning cyclic ready-queue turns bounded by a quantum.

C - Incorrect: SJF is a scheduling policy selecting the shortest next burst among ready jobs.

D - Correct: BST is a data structure that can support queue operations rather than a CPU scheduling policy.

Primary note proof

Study-note section OS19

> A binary search tree is a data structure that might implement a queue; it is not itself a CPU selection policy.

Study-note section OS15

> FCFS runs ready bursts in arrival order and ordinarily does not preempt.

Study-note section OS16

> Non-preemptive SJF/SJN chooses the shortest next burst among currently ready jobs.

Study-note section OS18

> RR serves a ready queue cyclically for at most quantum q per turn.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 23 | lines 12-17

7.1 First Come First Serve (FCFS)
• Jobs are executed on first come, first serve basis.
• Easy to understand and implement.
• Poor in performance as average wait time is high.
Wait time of each process is following
Process Wait Time: Service Time - Arrival Time

Operating System module | PDF page 25 | lines 10-14

7.4 Round Robin Scheduling
• Each process is provided a fix time to execute called quantum.
• Once a process is executed for given time period. Process is preempted and other process executes for given time period.
• Context switching is used to save states of preempted processes.

Operating System module | PDF page 23 | lines 23-26

7.2 Shortest Job First (SJF)
• Best approach to minimize waiting time.
• Impossible to implement
• Processer should know in advance how much time process will take.

Question sources

Operating Systems practice questions · Question 4

Question 46 | Q01287

Which of the following is not a type of file system?

A. FAT32

B. NTFS

C. EXT4

D. HTTP

Hint

Distinguish a file operation, a resource handle, an offset and file metadata.

Answer D: HTTP is an application communication protocol.

OS36 explicitly names FAT32, NTFS and ext4 as filesystem formats and contrasts HTTP as an application communication protocol. Filesystems organize stored file content and metadata; a communication protocol does not become a filesystem merely because it transfers file contents. Therefore HTTP is the single option outside the stated filesystem-format category.

Why each choice is right or wrong

A - Incorrect: FAT32 is explicitly identified as a filesystem format, making it a member of the requested category.

B - Incorrect: NTFS is explicitly identified as a filesystem format, not a communication-protocol exclusion.

C - Incorrect: ext4 is explicitly identified as a filesystem format in the complete study note.

D - Correct: HTTP is an application communication protocol rather than a filesystem format for organizing stored files.

Primary note proof

Study-note section OS36

> Sockets also support ordinary read/write, alongside socket/connect/send operations, so read is not uniquely “not a network call.” FAT32, NTFS and ext4 are filesystem formats; HTTP is an application communication protocol.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 48 | lines 5-8

> 13.1 File
> A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks. In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the file’s creator and user.

Question sources

Operating Systems practice questions · Question 5

Question 47 | Q01288

What is the purpose of a device driver in an operating system?

A. To manage user accounts

B. To manage hardware resources

C. To manage software applications

D. To manage network connections

Hint

Separate device hardware, device-specific software and general event or transfer mechanisms.

Answer B: To manage hardware resources

A driver implements I/O behavior for a device class and hides hardware peculiarities (PDF pp10,47). B is the closest option when hardware resources means its particular device; it should not imply one driver manages every resource.

Why each choice is right or wrong

A - Incorrect: Account administration is not the device-specific I/O job.

B - Correct: Device hardware I/O management is supported (pp10,47).

C - Incorrect: Applications use driver services but the driver does not manage applications generally.

D - Incorrect: Only a network-device driver concerns a network adapter; that is not every driver's purpose.

Primary note proof

Study-note section OS34

> A controller is device hardware exposing status, control and data registers. A driver converts OS requests into device-specific operations. Device independence and uniform naming hide brand-specific details; error handling reports/retries supported failures.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 30-37

> I/O subsystem comprised of I/O devices and their corresponding driver software. Drivers hides the peculiarities of specific hardware devices from the user as the device driver knows the peculiarities of the specific device.
> Operating System manages the communication between user and device drivers.
> The major activities of an operating system with respect to I/O Operation.
> • I/O operation means read or write operation with any file or any specific I/O device.
> • Program may require any I/O device while running.
> • Operating system provides the access to the required I/O device when required.

Operating System module | PDF page 47 | lines 19-26

> 12.5 Device driver
> Device driver is a program or routine developed for an I/O device. A device driver implements I/O operations or behaviors on a specific class of devices. For example a system supports one or a number of multiple brands of terminals, all slightly different terminals may have a single terminal driver. In the layered structure of I/O system, device driver lies between interrupt handler and device independent I/O software. The job of a device driver is following.
> To accept request from the device independent software above it.
> To see to it that the request is executed.

Question sources

Operating Systems practice questions · Question 6

Question 48 | Q01289

Which listed forms can serve as user interfaces to an operating system? Treat possible interface forms generally, rather than features of a particular product. Select all that apply.

A. Command-line interface (CLI)

B. Graphical user interface (GUI)

C. Natural language interface (NLI)

D. Virtual reality interface (VRI)

Hint

Distinguish identity checks, permission checks and the particular security property at issue.

Answer A, B, C, D: Command-line interface (CLI); Graphical user interface (GUI); Natural language interface (NLI); Virtual reality interface (VRI)

Command lines, graphical interfaces, speech or natural-language interaction, and virtual-reality interaction are possible interface forms. An OS need not implement all of them for each form to count as a possible interface. Because the stated question asks about capability in general rather than a particular product's installed features, all four listed forms satisfy it.

Why each choice is right or wrong

A - Correct: A CLI accepts text commands as a user interface.

B - Correct: A GUI provides graphical controls for user interaction.

C - Correct: Natural-language or speech interaction is a possible interface form.

D - Correct: VR interaction is also a possible interface form.

Primary note proof

Study-note section OS39

> CLI, GUI, speech/natural language and VR are possible interfaces, not a closed list excluding the last two.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 14 | lines 18-23

> Interactivity refers that a User is capable to interact with computer system. Operating system does the following activities related to interactivity.
> • OS provides user an interface to interact with system.
> • OS managers input devices to take inputs from the user. For example, keyboard.
> • OS manages output devices to show outputs to the user. For example, Monitor.
> • OS Response time needs to be short since the user submits and waits for the result.

Operating System module | PDF page 54 | lines 18-20

commands of the operating system. It can be used to do various types of operations, call application programs etc.
Security - Linux provides user security using authentication features like password

Question sources

Operating Systems practice questions · Question 7

Question 49 | Q01290

Which of the following is not a component of an operating system kernel?

A. Process scheduler

B. Memory manager

C. File system

D. Web browser

Hint

Distinguish command interfaces, privileged core services and the stages of booting.

Answer D: Web browser.

OS03 defines the kernel through core resource management, protection and system-call handling, while explicitly calling browsers applications. Scheduling, memory management and filesystem service are core resource-management functions in the textbook kernel sense. Some systems move services outside a kernel, but a web browser remains the clear application-level exclusion from the listed ordinary kernel responsibilities.

Why each choice is right or wrong

A - Incorrect: Process scheduling is a core resource-management function in the textbook kernel model used here.

B - Incorrect: Memory management is a core resource-management function rather than the listed user application.

C - Incorrect: Filesystem service belongs to ordinary OS resource management in this textbook kernel-level comparison.

D - Correct: The note explicitly identifies browsers as applications rather than components implementing the kernel's core role.

Primary note proof

Study-note section OS03

The kernel implements core resource management, protection and system-call handling. A shell reads/interprets user commands; a GUI is another interface. Browsers, office tools and editors are applications.

Study-note section OS01

An OS allocates CPU time, memory, storage, devices and communication resources, and provides services to programs.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 53 | lines 11-15

operating system. It is consisting of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.
System Library - System libraries are special functions or programs using which application programs or system utilities accesses Kernel's features. These libraries

Operating System module | PDF page 6 | lines 4-9

In multiprogramming environment, OS decides which process gets the processor when and how much time. This function is called process scheduling. Operating System does the following activities for processor management. Keeps tracks of processor and status of process. Program responsible for this task is known as traffic controller.
Allocates the processor (CPU) to a process.
De-allocates processor when processor is no longer required.

Operating System module | PDF page 31 | lines 24-31

Memory management is the functionality of an operating system which handles or manages primary memory. Memory management keeps track of each and every memory location either it is allocated to some process or it is free. It checks how much memory is to be allocated to processes. It decides which process will get memory at what time. It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status.
Memory management provides protection by using two registers, a base register and a limit register. The base register holds the smallest legal physical memory address and the limit register specifies the size of the range. For example, if the base register holds 300000 and the

Question sources

Operating Systems practice questions · Question 8

Question 50 | Q01291

Which of the following is not a type of system call used in operating systems?

A. Process control

B. File management

C. Network communication

D. Video editing

Hint

Identify the abstraction layer and separate a protected service request from an application task.

Answer D: Video editing is an application operation.

OS04 lists process control, file/device management, information retrieval and communication as service categories. It explicitly distinguishes video editing as an application operation. Network communication falls within communication services, so it is not excluded merely because it uses sockets rather than local files. The category outside generic system calls is video editing.

Why each choice is right or wrong

A - Incorrect: Process control is explicitly included among protected operating-system service categories.

B - Incorrect: File management is explicitly included among operating-system services requested through system calls.

C - Incorrect: Network communication falls within the communication service category rather than being excluded from system calls.

D - Correct: Video editing is an application-level operation, not a standard generic system-call service category.

Primary note proof

Study-note section OS04

> Process control, file/device management, information retrieval and communication are service categories. Video editing and a database-specific write_database() are application operations, not standard generic syscall categories/names.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 5-15

> An Operating System provides services to both the users and to the programs.
> • It provides programs, an environment to execute.
> • It provides users, services to execute the programs in a convenient manner.
> Operating Systems can provide the following common services:
> ✓ Program execution
> ✓ I/O operations
> ✓ File System manipulation
> ✓ Communication
> ✓ Error Detection
> ✓ Resource Allocation
> ✓ Protection

Operating System module | PDF page 10 | lines 23-28

> • Loads a program into memory.
> • Executes the program.

- Handles program's execution.
- Provides a mechanism for process synchronization.
- Provides a mechanism for process communication.
- Provides a mechanism for deadlock handling.

Operating System module | PDF page 11 | lines 3-8

- Program needs to read a file or write a file.
- The operating system gives the permission to the program for operation on file.
- Permission varies from read-only, read-write, denied and so on.
- Operating System provides an interface to the user to create/delete files.
- Operating System provides an interface to the user to create/delete directories.
- Operating System provides an interface to create the backup of file system.

Operating System module | PDF page 11 | lines 15-19

- Two processes often require data to be transferred between them.
- The both processes can be on the one computer or on different computer but are connected through computer network.
- Communication may be implemented by two methods either by Shared Memory or by Message Passing.

Question sources

Operating Systems practice questions · Question 9

Question 51 | Q01292

What is the purpose of a shell in an operating system?

A. To manage user accounts

B. To manage hardware resources

C. To manage software applications

D. To provide a user interface

Hint

Distinguish command interfaces, privileged core services and the stages of booting.

Answer D: To provide a user interface

PDF p54 calls the shell an interface to the kernel that takes user commands. D captures its interface role rather than an individual resource-management service.

Why each choice is right or wrong

A - Incorrect: Account management may be requested through shell commands, but is not its defining role.

B - Incorrect: Kernel provides low-level hardware services (p54).

C - Incorrect: Invoking applications is supported, but does not replace the general interface role.

D - Correct: Directly follows the user-command interface description (p54).

Primary note proof

Study-note section OS03

> The kernel implements core resource management, protection and system-call handling. A shell reads/interprets user commands; a GUI is another interface. Browsers, office tools and editors are applications.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 54 | lines 18-20

> commands of the operating system. It can be used to do various types of operations, call application programs etc.
> Security - Linux provides user security using authentication features like password

Operating System module | PDF page 54 | lines 24-30

> Hardware layer - Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc.).
> Kernel - Core component of Operating System, interacts directly with hardware, provides low level services to upper layer components.
> Shell - An interface to kernel, hiding complexity of kernel's functions from users. Takes commands from user and executes kernel's functions.
> Utilities - Utility programs giving user most of the functionalities of an operating systems.

Question sources

Operating Systems practice questions · Question 10

Question 52 | Q01293

Which listed systems are proprietary OS products rather than open-source systems as a whole? Interpret Linux as the open-source kernel and Android as AOSP without vendor additions. Select all that apply.

A. Linux

B. Android

C. macOS

D. Windows

Hint

Use the stated historical or platform scope and distinguish origin from technical capability.

Answer C, D: macOS; Windows

Windows and macOS are proprietary products. The existence of open-source components within macOS does not make the complete product open source. Linux is treated here as the open-source kernel, and Android is explicitly restricted to AOSP, excluding proprietary applications or services bundled by device vendors. With that component-versus-product scope fixed, both macOS and Windows belong in the selected set.

Why each choice is right or wrong

A - Incorrect: Linux is the open-source kernel under the stated scope.

B - Incorrect: AOSP is open source; vendor additions are explicitly excluded.

C - Correct: macOS is a proprietary product even though some components are open source.

D - Correct: Windows is a proprietary OS product.

Primary note proof

Study-note section OS40

> Linux is a Unix-compatible kernel written from scratch by Linus Torvalds; a distribution combines it with userland, libraries and tools. Android's base kernel layer is Linux; its framework and applications are higher layers. AOSP is open source, while a shipped device can include proprietary applications/services. Windows and macOS are both proprietary products, although macOS contains open-source components, so "which is not open source?" offering both is ambiguous. Microsoft Office is an application suite.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 53 | lines 5-7

> Linux is one of popular version of UNIX operating System. It is open source as its source code is freely available. It is free to use. Linux was designed considering UNIX compatibility. Its functionality list is quite similar to that of UNIX.

Operating System module | PDF page 54 | lines 5-17

> Portable - Portability means software can works on different types of hardware in same way. Linux kernel and application programs support their installation on any kind of hardware platform.
> Open Source - Linux source code is freely available and it is community-based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
> Multi-User - Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.
> Multiprogramming - Linux is a multiprogramming system means multiple applications can run at same time.
> Hierarchical File System - Linux provides a standard file structure in which system files/ user files are arranged.
> Shell - Linux provides a special interpreter program which can be used to execute

Question sources

Operating Systems practice questions · Question 11

Question 53 | Q01294

What is the purpose of a bootloader in an operating system?

A. To load the operating system kernel into memory

B. To manage hardware resources

C. To manage software applications

D. To provide a user interface

Hint

Distinguish command interfaces, privileged core services and the stages of booting.

Answer A: Load the OS kernel into memory.

OS03 places firmware/bootloader stages before transfer of control to a kernel loaded into RAM. This identifies loading the kernel as the bootloader purpose among the alternatives. Resource management is subsequently a kernel role, application management is a running-system function, and user interfaces are provided by shells/GUI components rather than defining the bootloader's purpose.

Why each choice is right or wrong

A - Correct: The bootloader prepares the kernel in memory and transfers control to it during the boot sequence.

B - Incorrect: Core hardware-resource management is the kernel's role after the bootloader hands over control.

C - Incorrect: Managing running applications belongs to the operating system's runtime services rather than defining the bootloader.

D - Incorrect: Shells and GUIs provide user interfaces; that is not the bootloader's principal role here.

Primary note proof

Study-note section OS03

> Firmware/bootloader stages execute before transferring control to a kernel loaded into RAM.

Study-note section OS03

> The kernel implements core resource management, protection and system-call handling. A shell reads/interprets user commands; a GUI is another interface. Browsers, office tools and editors are applications.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 53 | lines 11-15

> operating system. It is consisting of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.
> System Library - System libraries are special functions or programs using which application programs or system utilities accesses Kernel's features. These libraries

Operating System module | PDF page 32 | lines 11-14

> In dynamic loading, a routine of a program is not loaded until it is called by the program. All routines are kept on disk in a re-locatable load format. The main program is loaded into memory and is executed. Other routines methods or modules are loaded on request. Dynamic loading makes better memory space utilization and unused routines are never loaded.

Question sources

Operating Systems practice questions · Question 12

Question 54 | Q01295

What is the difference between a kernel-level thread and a user-level thread?

A. Kernel-level threads are faster than user-level threads

B. Kernel-level threads are managed by the operating system, while user-level threads are managed by the application

C. Kernel-level threads can access hardware resources directly, while user-level threads cannot

D. There is no difference between a kernel-level thread and a user-level thread

Hint

Compare per-thread execution context with shared process resources and the stated thread mapping.

Answer B: Kernel-level threads are managed by the operating system, while user-level threads are managed by the application

Application management of user threads is explicit on PDF p28; kernel management is explicit on p29. Kernel threads are generally slower to create/manage, and management location does not automatically grant application thread code kernel privilege.

Why each choice is right or wrong

A - Incorrect: Opposes the p29 slower-to-create/manage comparison.

B - Correct: Exactly matches management location on pp28-29.

C - Incorrect: Confuses scheduling/management by the kernel with execution privilege; p53 distinguishes user code from kernel code.

D - Incorrect: Management location and costs explicitly differ.

Primary note proof

Study-note section OS08

User-level threads are managed by a library; kernel threads are known to the OS. In many-to-one mapping, a blocking kernel call can block all user threads and one kernel execution context cannot use multiple CPUs simultaneously. One-to-one permits another kernel thread to run when one blocks. Many-to-many multiplexes user threads over several kernel threads. Speed and reliability depend on implementation; the management layer alone guarantees neither. Application names such as browser or Excel do not prove a particular version's threading model.

Study-note section OS-S02

In the conventional library-thread comparison, user-level thread creation and switching can avoid entering the kernel. Kernel-managed thread creation and management commonly incur additional kernel work and are therefore generally slower to create and manage in this comparison. This is a typical comparison, not a universal speed or reliability guarantee.

Study-note section OS-S02

Being scheduled or managed by the kernel does not give application thread code kernel privilege. Ordinary applications and supporting utilities execute in restricted user mode and request protected services through libraries and system calls. Kernel component code executes in privileged kernel mode and performs permitted low-level hardware operations.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 28 | lines 20-29

In this case, application manages thread management kernel is not aware of the existence of threads. The thread library contains code for creating and destroying threads, for passing message and data between threads, for scheduling thread execution and for saving and restoring thread contexts. The application begins with a single thread and begins running in that thread.
Advantages
• Thread switching does not require Kernel mode privileges.
• User level thread can run on any operating system.
• Scheduling can be application specific in the user level thread.
• User level threads are fast to create and manage.

Operating System module | PDF page 29 | lines 4-16

In this case, thread management done by the Kernel. There is no thread management code in the application area. Kernel threads are supported directly by the operating system. Any application can be programmed to be multithreaded. All of the threads within an application are supported within a single process.
The Kernel maintains context information for the process as a whole and for individuals' threads within the process. Scheduling by the Kernel is done on a thread basis. The Kernel performs thread creation, scheduling and management in Kernel space. Kernel threads are

generally slower to create and manage than the user threads.
Advantages
• Kernel can simultaneously schedule multiple threads from the same process on multiple processes.
• If one thread in a process is blocked, the Kernel can schedule another thread of the same process.

Operating System module | PDF page 53 | lines 21-29

Kernel component code executes in a special privileged mode called kernel mode with full access to all resources of the computer. This code represents a single process, executes in single address space and do not require any context switch and hence is very efficient and fast. Kernel runs each process and provides system services to processes, provides protected access to hardware to processes.
Support code which is not required to run in kernel mode is in System Library. User programs and other system programs works in User Mode which has no access to system hardware and kernel code. User programs/ utilities use System libraries to access Kernel functions to get system's low level tasks.

Question sources

Operating Systems practice questions · Question 13

Question 55 | Q01296

Which listed item is a general data structure rather than a deadlock-specific avoidance method or dependency-graph analysis?

A. Banker's algorithm

B. Wait-for graph algorithm

C. Resource allocation graph algorithm

D. Binary search tree algorithm

Hint

Compute remaining needs and simulate feasible completions before deciding whether an allocation is safe.

Answer D: Binary search tree algorithm

Banker’s algorithm checks whether tentative allocations leave a safe completion sequence and is an avoidance method. Wait-for and resource-allocation graphs represent dependencies used in deadlock reasoning. A binary search tree is a general data structure and does not, by its name alone, specify deadlock handling. The broader category distinguishes the unrelated item without misclassifying avoidance or detection as strict prevention.

Why each choice is right or wrong

A - Incorrect: Banker is a deadlock-avoidance method, so it belongs to the stated handling category.

B - Incorrect: A wait-for graph directly represents process dependencies for deadlock analysis.

C - Incorrect: A resource-allocation graph represents resource requests and assignments for deadlock reasoning.

D - Correct: A binary search tree is a general search structure rather than a deadlock-specific method.

Primary note proof

Study-note section OS25

> Avoidance allows requests only if the resulting state remains safe. Banker requires maximum claims. Compute Need=Max−Allocation.

Study-note section OS23

> In a resource-allocation graph, P→R is a request and an instance R→P is an assignment. Follow these directions to form waits. No cycle implies no deadlock; with one instance of every resource type, a cycle implies deadlock. With multiple instances a cycle alone is insufficient: another instance may permit completion. A wait-for graph replaces single-instance resource nodes with P→Q when P waits for Q.

Study-note section OS19

> A binary search tree is a data structure that might implement a queue; it is not itself a CPU selection policy.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

> • Loads a program into memory.
> • Executes the program.
> • Handles program's execution.
> • Provides a mechanism for process synchronization.
> • Provides a mechanism for process communication.
> • Provides a mechanism for deadlock handling.

Question sources

Operating Systems practice questions · Question 14

Question 56 | Q01297

What is the general execution role of a process in an operating system, regardless of the application it runs?

A. To manage user accounts

B. To manage hardware resources

C. To execute a program instance with its own address space and resources

D. To provide a user interface

Hint

Distinguish stored instructions, an executing instance and its saved execution state.

Answer C: To execute a program instance with its own address space and resources

A process is an executing instance of a program, accompanied by an address space and resources. Its application may administer accounts, interact with hardware through services, or display a user interface, but none of those specialized jobs defines every process. The execution option expresses the general role. It also distinguishes a running instance from a program stored as instructions.

Why each choice is right or wrong

A - Incorrect: Account administration is only one possible application responsibility.

B - Incorrect: Hardware management is an OS responsibility rather than the general role of every process.

C - Correct: Executing a program instance with an address space and resources is the process abstraction.

D - Incorrect: Many processes run without providing a user interface.

Primary note proof

Study-note section OS05

> A program is stored instructions; a process is an executing instance with address space and resources. A process does not universally “manage applications” or “provide a user interface.” The PCB records identity, state, PC/registers, scheduling information, memory/accounting details and I/O/open-resource state.

Study-note section OS01

> An OS allocates CPU time, memory, storage, devices and communication resources, and provides services to programs.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 16 | lines 6-9

A process is a program in execution. The execution of a process must progress in a sequential fashion. Definition of process is following.
• A process is defined as an entity which represents the basic unit of work to be implemented in the system.

Operating System module | PDF page 5 | lines 6-13

An operating System (OS) is an intermediary between users and computer hardware. It provides users an environment in which a user can execute programs conveniently and efficiently. In technical terms, it is software which manages hardware. An operating System controls the allocation of resources and services such as memory, processors, devices and information.
1.1 Definition
An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.

Question sources

Operating Systems practice questions · Question 15

Question 57 | Q01298

Which of the following is not a type of process state in an operating system?

A. Ready

B. Running

C. Blocked

D. Sleeping Beauty

Hint

Decide whether the process is eligible, running or waiting, then identify the responsible queue or scheduling role.

Answer D: Sleeping Beauty

The state list on PDF p16 contains Ready, Running and Waiting. Interpret Blocked as the waiting-for-event state. Sleeping Beauty is not one of this module's five named states.

Why each choice is right or wrong

A - Incorrect: Ready is explicitly listed.

B - Incorrect: Running is explicitly listed.

C - Incorrect: Blocked is the conventional paraphrase of waiting for an event; Waiting is the module's exact term.

D - Correct: Not present in the five-state model, and not a defined state term.

Primary note proof

Study-note section OS06

New processes are created/admitted; ready means eligible for CPU but not currently running. Running uses a CPU. Waiting/blocked means unable to proceed until an event; terminated means execution has ended. Running→ready follows time-slice expiry or another preemption. Running→waiting follows blocking I/O. I/O completion normally gives waiting→ready, not a guaranteed immediate running state.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 16 | lines 29-39

1. New The process is being created.
2. Ready The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run.
3. Running Process instructions are being executed (i.e. The process that is currently being executed).
4. Waiting The process is waiting for some event to occur (such as the completion of an I/O operation).
5. Terminated The process has finished execution.

Question sources

Operating Systems practice questions · Question 16

Question 58 | Q01299

What is the difference between a process and a thread in an operating system?

A. A process is a unit of execution, while a thread is a unit of scheduling

B. A process is a unit of scheduling, while a thread is a unit of execution

C. There is no difference between a process and a thread

D. A process can have multiple threads, while a thread can only belong to one process

Hint

Compare per-thread execution context with shared process resources and the stated thread mapping.

Answer D: A process can have multiple threads, while a thread can only belong to one process

PDF p27 says every thread belongs to exactly one process and separate flows improve parallel execution; p29 supports multiple threads within one process. D directly expresses that relationship.

Why each choice is right or wrong

A - Incorrect: Processes and threads can both be scheduled (pp20,29); this is not an exclusive distinction.

B - Incorrect: Both may participate in scheduling and execution; reversal does not define their containment.

C - Incorrect: PDF p27 explicitly distinguishes heavy processes from light threads.

D - Correct: Directly follows the one-process-per-thread and multithreaded-process definitions (pp27,29).

Primary note proof

Study-note section OS08

> A thread has its own PC, registers and stack. Threads in a process share code, address space and usually open-file resources. A process can contain many threads; a thread belongs to one process at a time. Threads are not a group of processes and need not execute sequentially. Independent processes can exist; cooperating ones communicate/share data.

Study-note section OS08

> User-level threads are managed by a library; kernel threads are known to the OS. In many-to-one mapping, a blocking kernel call can block all user threads and one kernel execution context cannot use multiple CPUs simultaneously. One-to-one permits another kernel thread to run when one blocks. Many-to-many multiplexes user threads over several kernel threads.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 27 | lines 6-14

> A thread is a flow of execution through the process code, with its own program counter, system registers and stack. A thread is also called a light weight process. Threads provide a way to improve application performance through parallelism. Threads represent a software approach to improving performance of operating system by reducing the overhead thread is equivalent to a classical process.
> Each thread belongs to exactly one process and no thread can exist outside a process. Each thread represents a separate flow of control. Threads have been successfully used in implementing network servers and web server. They also provide a suitable foundation for parallel execution of applications on shared memory multiprocessors. Following figure shows

Operating System module | PDF page 29 | lines 4-16

> In this case, thread management done by the Kernel. There is no thread management code in the application area. Kernel threads are supported directly by the operating system. Any application can be programmed to be multithreaded. All of the threads within an application are supported within a single process.
> The Kernel maintains context information for the process as a whole and for individuals' threads within the process. Scheduling by the Kernel is done on a thread basis. The Kernel performs thread creation, scheduling and management in Kernel space. Kernel threads are generally slower to create and manage than the user threads.

Advantages
• Kernel can simultaneously schedule multiple threads from the same process on multiple processes.
• If one thread in a process is blocked, the Kernel can schedule another thread of the same process.

Operating System module | PDF page 20 | lines 15-21

It is also called CPU scheduler. Main objective is increasing system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. CPU scheduler selects process among the processes that are ready to execute and allocates CPU to one of them.
Short term scheduler also known as dispatcher, execute most frequently and makes the fine-grained decision of which process to execute next. Short term scheduler is faster than long term scheduler.

Question sources

Operating Systems practice questions · Question 17

Question 59 | Q01300

Which of the following is not a type of synchronization mechanism used in operating systems?

A. Mutex

B. Semaphore

C. Monitor

D. Keyboard

Hint

Track available permits and blocked operations using the stated semaphore model.

Answer D: Keyboard.

OS10 describes monitors as protected operations with condition synchronization, and OS11 explains mutex/semaphore exclusion mechanisms. These coordinate concurrent execution. A keyboard is an input device rather than such a coordination abstraction. Interrupts generated by hardware may participate in event handling, but that does not make the keyboard itself a mutex, semaphore or monitor mechanism.

Why each choice is right or wrong

A - Incorrect: A mutex provides owned mutual exclusion and is a synchronization mechanism in the note.

B - Incorrect: A semaphore coordinates participants through atomic wait and signal operations, so it is a synchronization mechanism.

C - Incorrect: A monitor combines protected operations and condition synchronization, making it a genuine synchronization abstraction.

D - Correct: A keyboard is a physical input device rather than a process/thread synchronization abstraction.

Primary note proof

Study-note section OS11

> A semaphore provides atomic wait/P/down and signal/V/up operations. Under the nonnegative-token model, wait consumes one available token or blocks when none is available; signal supplies a token and may wake a waiter. Implementations using negative internal counts encode waiters differently, so state the model. A binary semaphore represents one permit; a counting semaphore tracks several. A mutex normally has ownership rules that a general semaphore lacks.

Study-note section OS10

> A barrier waits until a defined set of participants reaches a point; a monitor encapsulates protected operations with condition synchronization.

Study-note section OS34

> Input/output/storage/NIC devices are OS-managed hardware; user accounts are identities.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

> • Loads a program into memory.
> • Executes the program.
> • Handles program's execution.
> • Provides a mechanism for process synchronization.
> • Provides a mechanism for process communication.
> • Provides a mechanism for deadlock handling.

Operating System module | PDF page 43 | lines 7-10

> Computers operate on many kinds of devices. General types include storage devices (disks, tapes), transmission devices (network cards, modems), and human-interface devices (screen, keyboard, mouse). Other devices are more specialized. A device communicates with a computer system by sending signals over a cable or even through the air.

Question sources

Operating Systems practice questions · Question 18

Question 60 | Q01301

What is the purpose of a context switch in an operating system?

A. To save the state of a process or thread and switch to another process or thread

B. To manage hardware resources

C. To manage software applications

D. To provide a user interface

Hint

Distinguish stored instructions, an executing instance and its saved execution state.

Answer A: To save the state of a process or thread and switch to another process or thread

PDF p21 defines saving/restoring CPU context so execution can resume, enabling multiple processes to share a CPU. A identifies that specific operation.

Why each choice is right or wrong

A - Correct: Directly states save state and switch execution (p21).

B - Incorrect: Resource management is the wider OS purpose, not the specific state operation.

C - Incorrect: Application management is broader and omits saved execution state.

D - Incorrect: A user interface is not what a context switch provides.

Primary note proof

Study-note section OS05

> A context switch saves one execution context and restores another; useful application work is interrupted by this overhead.

Study-note section OS08

> A thread has its own PC, registers and stack.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 21 | lines 41-50

> A context switch is the mechanism to store and restore the state or context of a CPU in Process Control block so that a process execution can be resumed from the same point at a later time. Using this technique, a context switcher enables multiple processes to share a single CPU. Context switching is an essential part of a multitasking operating system features.
> When the scheduler switches the CPU from executing one process to execute another, the context switcher saves the content of all processor registers for the process being removed from the CPU, in its process descriptor. The context of a process is represented in the

process control block of a process. Context switch time is pure overhead. Context switching can significantly affect performance as modern computers have a lot of general

Question sources

Operating Systems practice questions · Question 19

Question 61 | Q01302

Which item is an application-layer protocol rather than a primitive OS IPC mechanism such as a stream, shared region or message queue?

A. Pipes

B. Shared memory

C. Message queues

D. HTTP requests

Hint

Identify how data moves, which participant waits and what event allows the operation to continue.

Answer D: HTTP requests

Pipes provide streams, shared memory provides a common region, and message queues preserve messages. These are primitive IPC mechanisms in the stated classification. HTTP is an application protocol that can use an underlying communication mechanism such as sockets. It can certainly carry interprocess communication, but it is the only listed application-layer protocol rather than one of the primitive mechanisms requested here.

Why each choice is right or wrong

A - Incorrect: Pipes are primitive stream-based IPC mechanisms.

B - Incorrect: Shared memory is a primitive common-region IPC mechanism.

C - Incorrect: Message queues are primitive message-preserving IPC mechanisms.

D - Correct: HTTP requests use an application-layer protocol over an underlying mechanism.

Primary note proof

Study-note section OS09

Shared memory lets processes access a common region and requires synchronization for conflicting updates. Message passing exchanges data through send/receive operations. Pipes provide streams; message queues preserve messages; sockets connect endpoints; RPC requests a remote operation. HTTP is an application protocol over a communication mechanism and can carry interprocess communication, so it is not universally “not IPC.”

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 11 | lines 15-19

> • Two processes often require data to be transferred between them.
> • The both processes can be on the one computer or on different computer but are connected through computer network.
> • Communication may be implemented by two methods either by Shared Memory or by Message Passing.

Question sources

Operating Systems practice questions · Question 20

Question 62 | Q01303

What is the general purpose of a system call in an operating system?

A. To manage user accounts

B. To request a protected kernel service

C. To manage software applications

D. To provide a user interface

Hint

Identify the abstraction layer and separate a protected service request from an application task.

Answer B: To request a protected kernel service

A system call is the mechanism through which executing code requests a protected kernel service. Process creation, file reads and socket operations are examples. A program that manages accounts, applications or a user interface may use system calls, but those tasks are not the mechanism’s general purpose. The option captures the common role without restricting it to one kind of resource.

Why each choice is right or wrong

A - Incorrect: Account management can use system calls but does not define their general purpose.

B - Correct: Requesting a protected kernel service states the general purpose.

C - Incorrect: Managing applications is a possible use rather than the system-call mechanism’s definition.

D - Incorrect: A UI may issue calls, but providing a UI is not the general system-call role.

Primary note proof

Study-note section OS04

A system call requests a protected kernel service, such as creating a process, reading a file or opening a socket. A user-facing API/library may wrap that request; not every library function enters the kernel.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 53 | lines 16-19

implement most of the functionalities of the operating system and do not requires kernel module's code access rights.
System Utility - System Utility programs are responsible to do specialized, individual level tasks.

Operating System module | PDF page 45 | lines 28-33

Application I/O Interface represents the structuring techniques and interfaces for the operating system to enable I/O devices to be treated in a standard, uniform way. The actual differences lies kernel level modules called device drivers which are custom tailored to corresponding devices but show one of the standard interfaces to applications. The purpose of the device-driver layer is to hide the differences among device controllers from the I/O subsystem of the kernel, such as the I/O system calls. Following are the characteristics of I/O

Question sources

Operating Systems practice questions · Question 21

Question 63 | Q01305

Which of the following is not a type of scheduling policy used in operating systems?

A. Preemptive

B. Non-preemptive

C. Cooperative

D. Infinite

Hint

Check whether the running burst can be interrupted before completion or blocking.

Answer D: Infinite is not a scheduling-policy category.

OS14 distinguishes preemptive and non-preemptive scheduling and explains cooperative scheduling as relying on voluntary release. OS19 explicitly rejects Infinite as a standard scheduling-policy category. Thus the first three labels describe genuine scheduling approaches, whereas infinite does not identify a selection or preemption rule. It should not be confused with a possible unbounded waiting-time outcome.

Why each choice is right or wrong

A - Incorrect: Preemptive scheduling can interrupt running work and return it to the ready state.

B - Incorrect: Non-preemptive scheduling retains a running burst until completion, blocking or voluntary release.

C - Incorrect: Cooperative scheduling is the recognized approach relying on voluntary release of execution.

D - Correct: Infinite names no scheduling selection or preemption policy in the provided note.

Primary note proof

Study-note section OS14

> Non-preemptive scheduling keeps a running CPU burst until completion, blocking or voluntary yielding. Preemptive scheduling can return running work to ready, usually after a timer quantum or a more urgent arrival. FCFS and non-preemptive SJF are non-preemptive; RR and SRTF are preemptive. Priority can be either, so its name alone cannot decide. Cooperative describes relying on voluntary release.

Study-note section OS19

> “Infinite” is not a standard scheduling-policy category.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 25 | lines 10-14

> 7.4 Round Robin Scheduling
> • Each process is provided a fix time to execute called quantum.
> • Once a process is executed for given time period. Process is preempted and other process executes for given time period.
> • Context switching is used to save states of preempted processes.

Operating System module | PDF page 23 | lines 12-17

> 7.1 First Come First Serve (FCFS)
> • Jobs are executed on first come, first serve basis.
> • Easy to understand and implement.
> • Poor in performance as average wait time is high.
> Wait time of each process is following
> Process Wait Time: Service Time - Arrival Time

Question sources

Operating Systems practice questions · Question 23

Question 64 | Q01306

24.What is the purpose of a page table in memory management?

A. To translate virtual addresses to physical addresses

B. To manage user accounts

C. To manage software applications

D. To provide a user interface

Hint

Split the address into page and offset, then apply the mapping and access checks.

Answer A: To translate virtual addresses to physical addresses

Page number indexes a table containing physical page bases (PDF p34); combining the base with the page offset yields the physical address (p35). Thus a page table participates in virtual-to-physical translation, A.

Why each choice is right or wrong

A - Correct: Matches page-table address lookup (pp34-35).

B - Incorrect: User accounts are not the page-number-to-frame mapping.

C - Incorrect: Application management is broader than the translation data structure.

D - Incorrect: An execution address map does not provide the user interface.

Primary note proof

Study-note section OS30

> Pages and physical frames have equal fixed size P. Divide virtual address v into page=floor(v/P) and offset=v mod P. The page table maps a page to a frame and protection/status flags. Physical address=frame×P+offset. A TLB caches translations; a TLB miss may require a table walk while the page is already resident.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 34 | lines 47-56

> External fragmentation is avoided by using paging technique. Paging is a technique in which physical memory is broken into blocks of the same size called pages (size is power of 2, between 512 bytes and 8192 bytes). When a process is to be executed, it's corresponding pages are loaded into any available memory frames.
> Logical address space of a process can be non-contiguous and a process is allocated physical memory whenever the free memory frame is available. Operating system keeps track of all free frames. Operating system needs n free frames to run a program of size n pages.
> Address generated by CPU is divided into
> • Page number (p) -- page number is used as an index into a page table which contains base address of each page in physical memory.

Operating System module | PDF page 33 | lines 6-13

> The run-time mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device. MMU uses following mechanism to convert virtual address to physical address.
> • The value in the base register is added to every address generated by a user process which is treated as offset at the time it is sent to memory. For example, if the base register value is 10000, then an attempt by the user to use address location 100 will be dynamically reallocated to location 10100.
> • The user program deals with virtual addresses; it never sees the real physical addresses.

Operating System module | PDF page 35 | lines 3-4

> • Page offset (d) -- page offset is combined with base address to define the physical memory address.

Question sources

Operating Systems practice questions · Question 24

Question 65 | Q01308

What is the purpose of a system monitor in an operating system?

A. To manage user accounts

B. To manage hardware resources

C. To manage software applications

D. To provide information about system performance and resource usage

Hint

Identify whether data is in transit, retained for reuse or queued for a device.

Answer D: Report performance and resource usage.

OS35 says a system monitor reports CPU, memory, storage and network usage and measures rather than creating more resources. This matches providing information about performance and usage. User-account administration, hardware allocation and application management are broader OS/administration functions, not the monitor's defining reporting role. Observing a resource is distinct from allocating or increasing it.

Why each choice is right or wrong

A - Incorrect: Account management is an administration function, not the monitor's specified performance-reporting purpose.

B - Incorrect: The monitor measures hardware usage rather than serving as the general resource-allocation manager.

C - Incorrect: Application management is broader than the monitor's role of reporting system performance and consumption.

D - Correct: Reporting system performance and resource usage directly matches the monitor definition in the note.

Primary note proof

Study-note section OS35

> A system monitor reports CPU, memory, storage and network usage; it measures rather than directly creating more resources.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 6 | lines 31-35

> Control over system performance -- Recording delays between request for a service and response from the system.
> Job accounting -- Keeping track of time and resources used by various jobs and users.
> Error detecting aids -- Production of dumps, traces, error messages and other debugging and error detecting aids.

Question sources

Operating Systems practice questions · Question 26

Question 66 | Q01309

What is the difference between a kernel and a shell in an operating system?

A. A kernel is the core of the operating system, while a shell provides a user interface

B. A kernel provides a user interface, while a shell manages hardware resources

C. There is no difference between a kernel and a shell

D. A kernel manages hardware resources, while a shell manages software applications

Hint

Distinguish command interfaces, privileged core services and the stages of booting.

Answer A: A kernel is the core of the operating system, while a shell provides a user interface

The kernel is the core interacting with hardware (PDF p53); the shell takes user commands and interfaces to the kernel (p54). A preserves both roles.

Why each choice is right or wrong

A - Correct: Both halves match pp53-54.

B - Incorrect: Reverses the explicit roles.

C - Incorrect: The module lists and distinguishes the two layers.

D - Incorrect: First half is supported; invoking applications is only part of shell use, and omits its defining command interface.

Primary note proof

Study-note section OS03

> The kernel implements core resource management, protection and system-call handling. A shell reads/interprets user commands; a GUI is another interface. Browsers, office tools and editors are applications.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 53 | lines 11-15

> operating system. It is consisting of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.
> System Library - System libraries are special functions or programs using which application programs or system utilities accesses Kernel's features. These libraries

Operating System module | PDF page 54 | lines 18-20

> commands of the operating system. It can be used to do various types of operations, call application programs etc.
> Security - Linux provides user security using authentication features like password

Operating System module | PDF page 54 | lines 24-30

> Hardware layer - Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc.).
> Kernel - Core component of Operating System, interacts directly with hardware, provides low level services to upper layer components.
> Shell - An interface to kernel, hiding complexity of kernel's functions from users. Takes commands from user and executes kernel's functions.
> Utilities - Utility programs giving user most of the functionalities of an operating systems.

Question sources

Operating Systems practice questions · Question 27

Question 67 | Q01310

Which of the following is not a type of system call used in file management?

A. open()

B. close()

C. read()

D. write_database()

Hint

Distinguish a file operation, a resource handle, an offset and file metadata.

Answer D: write_database() is not a standard generic file-management system call.

OS36 identifies open, close and read as ordinary file operations, and OS04 explicitly distinguishes database-specific write_database() as an application operation rather than a standard generic syscall name. A database operation may internally use actual file syscalls, but that composition does not make its application-specific name a generic kernel file-management call.

Why each choice is right or wrong

A - Incorrect: open obtains an open-file reference and is a standard file-management operation described here.

B - Incorrect: close releases an open-file reference and is a standard file-management operation.

C - Incorrect: read transfers bytes from an open resource and is a standard generic file operation.

D - Correct: write_database is an application-specific database operation rather than a standard generic file-management syscall.

Primary note proof

Study-note section OS36

> A file has content and metadata such as name, size, owner, permissions and timestamps. open obtains an open-file reference, read/write transfer bytes, seek changes an offset, close releases a reference, truncate changes size and unlink removes a name.

Study-note section OS04

> Process control, file/device management, information retrieval and communication are service categories. Video editing and a database-specific write_database() are application operations, not standard generic syscall categories/names.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 11 | lines 3-8

> • Program needs to read a file or write a file.
> • The operating system gives the permission to the program for operation on file.
> • Permission varies from read-only, read-write, denied and so on.
> • Operating System provides an interface to the user to create/delete files.
> • Operating System provides an interface to the user to create/delete directories.
> • Operating System provides an interface to create the backup of file system.

Question sources

Operating Systems practice questions · Question 28

Question 68 | Q01311

Which item is a hardware/CPU event-notification mechanism rather than a high-level lock, participant rendezvous, or protected-operation abstraction?

A. Locks

B. Barriers

C. Monitors

D. Interrupts

Hint

Identify conflicting shared accesses and the safety or progress property required.

Answer D: Interrupts

Locks coordinate entry to protected state, barriers wait for participants to reach a common point, and monitors encapsulate protected operations with conditions. Interrupts notify the CPU of events and may help coordinate work, but belong to a different abstraction level. The classification therefore selects interrupts without denying that interrupts can participate in synchronization or completion notification.

Why each choice is right or wrong

A - Incorrect: A lock is a high-level mutual-exclusion abstraction.

B - Incorrect: A barrier is a participant rendezvous abstraction.

C - Incorrect: A monitor encapsulates protected operations and condition synchronization.

D - Correct: An interrupt is a CPU event-notification mechanism, the different abstraction level requested.

Primary note proof

Study-note section OS10

> Keep critical regions small and ensure every conflicting access uses the same protection. A barrier waits until a defined set of participants reaches a point; a monitor encapsulates protected operations with condition synchronization.

Study-note section OS11

> A mutex normally has ownership rules that a general semaphore lacks.

Study-note section OS34

> Polling checks readiness repeatedly; interrupts request CPU attention; DMA transfers blocks after setup without CPU work for every byte.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

• Loads a program into memory.
• Executes the program.
• Handles program's execution.
• Provides a mechanism for process synchronization.
• Provides a mechanism for process communication.
• Provides a mechanism for deadlock handling.

Operating System module | PDF page 45 | lines 10-22

The CPU hardware uses an interrupt request line wire which helps CPU to sense after executing every instruction. When the CPU checks that a controller has put a signal on the interrupt request line, the CPU saves a state, such as the current value of the instruction pointer, and jumps to the interrupt handler routine at a fixed address. The interrupt handler part determines the cause of the interrupt performs the necessary processing and executes an interrupt instruction to return the CPU to its execution state.
The basic mechanism of interrupt enables the CPU to respond to an asynchronous event, such as when a device controller becomes ready for service. Most CPUs have two interrupt request lines.
• non-maskable interrupt - Such kind of interrupts are reserved for events like unrecoverable memory errors.
• maskable interrupt - Such interrupts can be switched off by the CPU before the execution of critical instructions that must not be interrupted.

Question sources

Operating Systems practice questions · Question 29

Question 69 | Q01312

What is the purpose of a virtual machine in an operating system?

A. To manage user accounts

B. To manage hardware resources

C. To manage software applications

D. To provide a virtualized environment for running multiple operating systems on a single physical machine

Hint

Identify the abstraction layer and separate a protected service request from an application task.

Answer D: A virtualized environment can run guest operating systems on one physical host.

OS04 defines virtualization as presenting a virtual machine environment, often allowing guest operating systems on one host through a hypervisor. That is exactly the complete description in D. Account, hardware and application management may occur within host or guest systems, but those broad administrative roles do not define why a virtual machine is provided.

Why each choice is right or wrong

A - Incorrect: User-account management is not the defining purpose of presenting a virtual machine environment.

B - Incorrect: Hardware is mediated, but general hardware management alone omits the guest-machine environment requested.

C - Incorrect: Application management does not define a virtual machine or distinguish it from an ordinary running OS.

D - Correct: Providing guest-machine environments on one physical host directly matches the note's virtualization definition.

Primary note proof

Study-note section OS04

> Virtualization presents a virtual machine environment, often allowing guest operating systems on one host through a hypervisor. It differs from virtual memory, which translates a process's addresses. Kernel, shell, API and VM are not interchangeable labels.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 37 | lines 6-11

> Virtual memory is a technique that allows the execution of processes which are not completely available in memory. The main visible advantage of this scheme is that programs can be larger than physical memory. Virtual memory is the separation of user logical memory from physical memory.
> This separation allows an extremely large virtual memory to be provided for programmers when only a smaller physical memory is available. Following are the situations, when entire

Question sources

Operating Systems practice questions · Question 30

Question 70 | Q01313

Which item coordinates access but does not itself transfer arbitrary data payloads or request an operation in another process?

A. Message passing

B. Shared memory

C. Remote procedure call (RPC)

D. File locking

Hint

Identify how data moves, which participant waits and what event allows the operation to continue.

Answer D: File locking

Message passing transfers messages, shared memory carries data through a common region, and RPC requests an operation in another process or machine. File locking instead coordinates who may access a file or region. Locks can therefore be part of IPC synchronization while not being a payload-transfer mechanism themselves. The narrower wording makes file locking the unique match without denying its coordination role.

Why each choice is right or wrong

A - Incorrect: Message passing transfers data using send/receive operations.

B - Incorrect: Shared memory carries data in a common accessible region.

C - Incorrect: RPC communicates a request to perform another process's operation.

D - Correct: File locking coordinates access rather than carrying arbitrary payloads itself.

Primary note proof

Study-note section OS09

> Shared memory lets processes access a common region and requires synchronization for conflicting updates. Message passing exchanges data through send/receive operations. Pipes provide streams; message queues preserve messages; sockets connect endpoints; RPC requests a remote operation.

Study-note section OS09

> File locks coordinate access rather than transfer arbitrary payloads, though locking is synchronization within IPC.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 11 | lines 15-19

> • Two processes often require data to be transferred between them.
> • The both processes can be on the one computer or on different computer but are connected through computer network.
> • Communication may be implemented by two methods either by Shared Memory or by Message Passing.

Question sources

Operating Systems practice questions · Question 31

Question 71 | Q01314

What is the purpose of a file system in an operating system?

A. To manage user accounts

B. To manage hardware resources

C. To manage software applications

D. To manage files and directories on storage devices

Hint

Distinguish a file operation, a resource handle, an offset and file metadata.

Answer D: To manage files and directories on storage devices

The file system tracks file information and locations and organizes directories (PDF p6); files store information on secondary storage (p48). D captures these functions.

Why each choice is right or wrong

A - Incorrect: Account administration manages user identities and accounts, rather than organizing stored file contents and directory names.

B - Incorrect: Hardware-resource management spans processors, memory and devices; it is broader than the file system's file-and-directory responsibility.

C - Incorrect: Application management does not describe file/directory organization.

D - Correct: Matches locations, directory organization and secondary storage (pp6,48).

Primary note proof

Study-note section OS36

> A file has content and metadata such as name, size, owner, permissions and timestamps. open obtains an open-file reference, read/write transfer bytes, seek changes an offset, close releases a reference, truncate changes size and unlink removes a name.

Study-note section OS37

> A single-level directory requires unique names in one namespace. Two-level directories separate users. A tree allows nested directories and absolute/relative paths; links can create shared graphs. Modern general-purpose systems commonly present hierarchical namespaces, but a feature name alone does not prove that every system excludes simpler arrangements.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 6 | lines 19-23

> A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions. Operating System does the following activities for file management.

Keeps track of information, location, uses, status etc. The collective facilities are often known as file system.

Operating System module | PDF page 48 | lines 5-8

13.1 File
A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks. In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the file's creator and user.

Question sources

Operating Systems practice questions · Question 32

Question 72 | Q01315

In this question only, "task" means a thread within a process. Which statement describes the process-task ownership relationship?

A. There is no difference between a process and a task

B. A process is a unit of execution, while a task is a unit of scheduling

C. A process can have multiple tasks, while a task can only belong to one process

D. A task is a unit of execution, while a process is a unit of scheduling

Hint

Compare per-thread execution context with shared process resources and the stated thread mapping.

Answer C: A process can have multiple tasks, while a task can only belong to one process

With task explicitly defined to mean thread, a process may contain several tasks while each task belongs to one process at a time. The tasks share the process address space but have their own execution context. This ownership relationship does not follow from the word task universally; it follows from the definition supplied here. Execution-versus-scheduling slogans do not state that relationship.

Why each choice is right or wrong

A - Incorrect: A thread/task is not identical to its containing process under this definition.

B - Incorrect: The execution-versus-scheduling description does not express the defined ownership relationship.

C - Correct: A process can contain multiple threads/tasks, each belonging to one process at a time.

D - Incorrect: The reversed execution/scheduling labels do not describe thread ownership.

Primary note proof

Study-note section OS08

> A thread has its own PC, registers and stack. Threads in a process share code, address space and usually open-file resources. A process can contain many threads; a thread belongs to one process at a time. Threads are not a group of processes and need not execute sequentially. Independent processes can exist; cooperating ones communicate/share data.

Study-note section OS05

> Task is OS-specific terminology: it may mean process, thread or schedulable entity.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 16 | lines 6-9

> A process is a program in execution. The execution of a process must progress in a sequential fashion. Definition of process is following.
> • A process is defined as an entity which represents the basic unit of work to be implemented in the system.

Operating System module | PDF page 17 | lines 4-7

> Each process is represented in the operating system by a process control block (PCB) also called a task control block. PCB is the data structure used by the operating system. Operating system groups all information that needs about particular process. PCB contains many pieces of information associated with a specific process which is described below.

Question sources

Operating Systems practice questions · Question 33

Question 73 | Q01316

Which of the following is not a type of memory allocation algorithm used in operating systems?

A. First-Fit

B. Best-Fit

C. Next-Fit

D. Binary-Fit

Hint

Apply the stated hole-selection or base-and-bound rule with its explicit convention.

Answer D: Binary-Fit is not one of the standard fit-policy names.

OS28 defines first-fit, next-fit, best-fit and worst-fit, then explicitly says binary-fit is not one of these standard names. First-fit, best-fit and next-fit therefore belong to the allocation-policy category used by the question. Binary-fit is the supported exclusion; it should not be invented as a synonym for another algorithm merely because the word binary is familiar.

Why each choice is right or wrong

A - Incorrect: First-fit selects the first adequate hole and is an explicitly defined memory-allocation policy.

B - Incorrect: Best-fit chooses the smallest adequate hole and is an explicitly defined memory-allocation policy.

C - Incorrect: Next-fit resumes search from the previous position and is an explicitly defined allocation policy.

D - Correct: The note explicitly excludes binary-fit from the standard memory-allocation fit-policy names.

Primary note proof

Study-note section OS28

> First-fit selects the first adequate hole; next-fit continues from the previous search position; best-fit selects the smallest adequate hole; worst-fit selects the largest. No policy universally optimizes both search time and future storage utilization. Binary-fit is not one of these standard names.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 34 | lines 34-45

> External
> fragmentation
> Total memory space is enough to satisfy a request or to reside a
> process in it, but it is not contiguous so it cannot be used.
> 2
> Internal
> fragmentation
> Memory block assigned to process is bigger. Some portion of memory
> is left unused as it cannot be used by another process.
> External fragmentation can be reduced by compaction or shuffle memory contents to place all free memory together in one large block. To make compaction feasible, relocation should be dynamic

Question sources

Operating Systems practice questions · Question 34

Question 74 | Q01317

What is the purpose of a system call interface in an operating system?

A. To manage user accounts

B. To manage hardware resources

C. To manage software applications

D. To provide a standardized interface for application programs to make system calls

Hint

Identify the abstraction layer and separate a protected service request from an application task.

Answer D: Provide the interface through which applications request system calls.

OS04 defines a system call as requesting a protected kernel service and distinguishes an API/library wrapper from the direct boundary. Option D describes the interface that applications use to make those requests. Account, hardware and application management can be services reached through it; they are not the general purpose of the interface itself.

Why each choice is right or wrong

A - Incorrect: Account operations may be exposed as services, but account management does not define the interface itself.

B - Incorrect: Hardware operations may be requested through services; general hardware management is not the interface definition.

C - Incorrect: Application management is a possible service use rather than the purpose of the syscall interface.

D - Correct: The interface supplies the application-to-system-call request path for protected kernel services.

Primary note proof

Study-note section OS04

> A system call requests a protected kernel service, such as creating a process, reading a file or opening a socket. A user-facing API/library may wrap that request; not every library function enters the kernel. Distinguish the application API from the direct user/kernel boundary when describing access to OS services.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 45 | lines 28-33

> Application I/O Interface represents the structuring techniques and interfaces for the operating system to enable I/O devices to be treated in a standard, uniform way. The actual differences lies kernel level modules called device drivers which are custom tailored to corresponding devices but show one of the standard interfaces to applications. The purpose of

the device-driver layer is to hide the differences among device controllers from the I/O subsystem of the kernel, such as the I/O system calls. Following are the characteristics of I/O

Operating System module | PDF page 53 | lines 16-19

implement most of the functionalities of the operating system and do not requires kernel module's code access rights.
System Utility - System Utility programs are responsible to do specialized, individual level tasks.

Question sources

Operating Systems practice questions · Question 35

Question 75 | Q01319

What is the purpose of a dispatcher in an operating system?

A. To manage user accounts

B. To manage hardware resources

C. To manage software applications

D. To switch the context of execution from one process to another

Hint

Decide whether the process is eligible, running or waiting, then identify the responsible queue or scheduling role.

Answer D: To switch the context of execution from one process to another

PDF p19 describes the dispatcher transferring execution to a queued process; p21 defines the context save/restore mechanism. D is the specific execution handoff rather than a broad management service.

Why each choice is right or wrong

A - Incorrect: Account management does not identify execution handoff.

B - Incorrect: Hardware management is a broad OS responsibility.

C - Incorrect: Application management is not the specific handoff operation.

D - Correct: Switching saved execution contexts matches pp19,21.

Primary note proof

Study-note section OS06

Long-term scheduling admits jobs; medium-term scheduling suspends/swaps them; short-term scheduling selects ready work. The dispatcher performs the handoff/context restore after selection.

Study-note section OS05

A context switch saves one execution context and restores another; useful application work is interrupted by this overhead.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 19 | lines 20-25

Non-Running Processes that are not running are kept in queue, waiting for their turn to execute. Each entry in the queue is a pointer to a particular process. Queue is implemented by using linked list. Use of dispatcher is as follows. When a process is interrupted, that process is transferred in the waiting queue. If the process has completed or aborted, the process is discarded. In either case, the dispatcher then selects a process from the queue to execute.

Operating System module | PDF page 21 | lines 41-50

A context switch is the mechanism to store and restore the state or context of a CPU in Process Control block so that a process execution can be resumed from the same point at a later time. Using this technique, a context switcher enables multiple processes to share a single CPU. Context switching is an essential part of a multitasking operating system features.
When the scheduler switches the CPU from executing one process to execute another, the context switcher saves the content of all processor registers for the process being removed from the CPU, in its process descriptor. The context of a process is represented in the process control block of a process. Context switch time is pure overhead. Context switching can significantly affect performance as modern computers have a lot of general

Question sources

Operating Systems practice questions · Question 37

Question 76 | Q01320

Which listed item is not a hardware resource managed by an operating system?

A. CPU

B. Memory

C. Keyboard

D. User account

Hint

Separate broad resource-management goals from individual services or applications.

Answer D: User account

CPU time, memory and input devices such as a keyboard are hardware resources allocated or controlled by the OS. A user account is an identity used in administration and protection, rather than a hardware device. The OS may manage accounts too, but that does not put them in the hardware-resource category. Completing the stem and replacing one option gives a unique exclusion.

Why each choice is right or wrong

A - Incorrect: The CPU is a managed processing resource.

B - Incorrect: Memory is a managed storage resource used during execution.

C - Incorrect: A keyboard is a managed input device.

D - Correct: A user account is an identity rather than a hardware resource.

Primary note proof

Study-note section OS01

> An OS allocates CPU time, memory, storage, devices and communication resources, and provides services to programs.

Study-note section OS34

> Input/output/storage/NIC devices are OS-managed hardware; user accounts are identities.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 5 | lines 6-13

> An operating System (OS) is an intermediary between users and computer hardware. It provides users an environment in which a user can execute programs conveniently and efficiently. In technical terms, it is software which manages hardware. An operating System controls the allocation of resources and services such as memory, processors, devices and information.
> 1.1 Definition
> An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.

Operating System module | PDF page 43 | lines 7-10

> Computers operate on many kinds of devices. General types include storage devices (disks, tapes), transmission devices (network cards, modems), and human-interface devices (screen, keyboard, mouse). Other devices are more specialized. A device communicates with a computer system by sending signals over a cable or even through the air.

Question sources

Operating Systems practice questions · Question 38

Question 77 | Q01321

39.What is the difference between a process and a program in an operating system?

A. There is no difference between a process and a program

B. A process is a running instance of a program, while a program is a set of instructions

C. A program is a running instance of a process, while a process is a set of instructions

D. A program is a unit of execution, while a process is a unit of scheduling

Hint

Distinguish stored instructions, an executing instance and its saved execution state.

Answer B: A process is a running instance of a program, while a program is a set of instructions

PDF p16 directly contrasts a dynamic executing process with a static program containing processor instructions. B states that distinction.

Why each choice is right or wrong

A - Incorrect: Contradicted by the explicit static/dynamic contrast.

B - Correct: Matches the definitions on p16.

C - Incorrect: Reverses program and process.

D - Incorrect: Program is static instructions, not itself an executing unit (p16).

Primary note proof

Study-note section OS05

> A program is stored instructions; a process is an executing instance with address space and resources. A process does not universally “manage applications” or “provide a user interface.” The PCB records identity, state, PC/registers, scheduling information, memory/accounting details and I/O/open-resource state.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 16 | lines 6-9

> A process is a program in execution. The execution of a process must progress in a sequential fashion. Definition of process is following.
> • A process is defined as an entity which represents the basic unit of work to be implemented in the system.

Operating System module | PDF page 16 | lines 21-24

A program by itself is not a process. It is a static entity made up of program statement while process is a dynamic entity. Program contains the instructions to be executed by processor. A program takes a space at single place in main memory and continues to stay there. A program does not perform any action by itself.

Question sources

Operating Systems practice questions · Question 39

Question 78 | Q01322

Which listed operation can receive bytes from a socket while also serving as a generic file-input operation in the descriptor model?

A. socket()

B. connect()

C. read()

D. send()

Hint

Distinguish a file operation, a resource handle, an offset and file metadata.

Answer C: read()

Sockets can support the ordinary read/write interface as well as specialized socket operations. A read therefore need not be confined to disk files: it can receive bytes from a socket descriptor. socket creates an endpoint, connect establishes a connection, and send transmits data. Only read matches both the generic file-input role and the ability to receive socket data in this model.

Why each choice is right or wrong

A - Incorrect: socket creates a communication endpoint rather than reading bytes from it.

B - Incorrect: connect establishes a connection rather than receiving its data.

C - Correct: read is generic descriptor input and can receive bytes from a socket.

D - Incorrect: send transmits data rather than performing the requested input operation.

Primary note proof

Study-note section OS36

Sockets also support ordinary read/write, alongside socket/connect/send operations, so read is not uniquely “not a network call.” FAT32, NTFS and ext4 are filesystem formats; HTTP is an application communication protocol.

Study-note section OS-S04

In the descriptor-based socket API, socket() creates a communication endpoint and returns its descriptor. connect() associates the socket with the requested peer address; send() transmits data

to another socket. Receiving through read() uses the ordinary descriptor-input interface described in the file-services lesson.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 11 | lines 15-19

• Two processes often require data to be transferred between them.
• The both processes can be on the one computer or on different computer but are connected through computer network.
• Communication may be implemented by two methods either by Shared Memory or by Message Passing.

Operating System module | PDF page 11 | lines 3-8

• Program needs to read a file or write a file.
• The operating system gives the permission to the program for operation on file.
• Permission varies from read-only, read-write, denied and so on.
• Operating System provides an interface to the user to create/delete files.
• Operating System provides an interface to the user to create/delete directories.
• Operating System provides an interface to create the backup of file system.

Question sources

Operating Systems practice questions · Question 40

Question 79 | Q01323

What is the purpose of a system clock in an operating system?

A. To manage user accounts

B. To manage hardware resources

C. To manage software applications

D. To provide a reference for measuring time and scheduling processes and events

Hint

Identify whether data is in transit, retained for reuse or queued for a device.

Answer D: To provide a reference for measuring time and scheduling processes and events

The clock driver maintains time and decrements the quantum counter to invoke the scheduler (PDF p46, section 12.3 Clocks). D is the matching time-and-scheduling purpose.

Why each choice is right or wrong

A - Incorrect: Accounts are not the clock's task.

B - Incorrect: Generic resource management omits the clock's time reference.

C - Incorrect: Software management is not its stated task.

D - Correct: Time of day and quantum expiration are both clock-driver functions on p46.

Primary note proof

Study-note section OS35

> The clock/timer supports timekeeping, timeouts, accounting and preemptive scheduling.

Study-note section OS-S04

> IRQ means interrupt request. A controller can raise an interrupt to request CPU attention. In a clock-tick scheduler, the scheduler initializes a running process's quantum counter; each clock interrupt decrements it, and reaching zero triggers scheduling. Clock support also maintains time of day and usage accounting.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 46 | lines 19-33

> Maintaining the time
> of the day
> The clock driver implements the time of day or the real time
> clock function. It requires incrementing a counter at each
> clock tick.
> 2
> Preventing processes
> from running too long
> As a process is started, the scheduler initializes the quantum
> counter in clock ticks for the process. The clock driver
> decrements the quantum counter by 1, at every clock
> interrupts. When the counter gets to zero, clock driver calls
> the scheduler to set up another process. Thus clock driver
> helps in preventing processes from running longer than time
> slice allowed.

Question sources

Operating Systems practice questions · Question 41

Question 80 | Q01324

42.Which of the following is not a type of device managed by an operating system?

A. Input devices

B. Output devices

C. Storage devices

D. User accounts

Hint

Separate device hardware, device-specific software and general event or transfer mechanisms.

Answer D: User accounts

PDF p43 lists storage, transmission and human-interface hardware. User accounts concern authentication (p50), rather than physical I/O device categories.

Why each choice is right or wrong

A - Incorrect: Human input devices are listed (p43).

B - Incorrect: Screens and related human-interface devices are listed (p43).

C - Incorrect: Disks and tapes are storage devices (p43).

D - Correct: A user identity/account is authenticated, not a physical device (p50).

Primary note proof

Study-note section OS34

> Input/output/storage/NIC devices are OS-managed hardware; user accounts are identities.

Study-note section OS39

> Authentication verifies identity; authorization checks permitted operations.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 43 | lines 7-10

> Computers operate on many kinds of devices. General types include storage devices (disks, tapes), transmission devices (network cards, modems), and human-interface devices (screen, keyboard, mouse). Other devices are more specialized. A device communicates with a computer system by sending signals over a cable or even through the air.

Operating System module | PDF page 50 | lines 18-28

> Authentication refers to identifying each user of the system and associating the executing programs with those users. It is the responsibility of the Operating System to create a protection system which ensures that a user who is running a particular program is authentic. Operating Systems generally identifies/authenticates users using following three ways:
> Username / Password - User need to enter a registered username and password with Operating system to login into the system.
> User card/key - User need to punch card in card slot, or enter key generated by key generator in option provided by operating system to login into the system.
> User attribute - fingerprint/ eye retina pattern/ signature - User need to pass his/her attribute via designated input device used by operating system to login into the system.

Question sources

Operating Systems practice questions · Question 42

Question 81 | Q01325

What is the purpose of a file descriptor in an operating system?

A. To manage user accounts

B. To manage hardware resources

C. To manage software applications

D. To provide a reference to an open file or communication endpoint

Hint

Distinguish a file operation, a resource handle, an offset and file metadata.

Answer D: A process-local reference to an open file or communication endpoint.

OS36 defines a file descriptor as a process-local integer handle and separately places offset and access mode in associated open-file state. It also describes sockets supporting ordinary descriptor I/O. The descriptor therefore refers to an open resource or endpoint; it is not an account manager, hardware manager or application manager. D is the specific role matching that definition.

Why each choice is right or wrong

A - Incorrect: A descriptor identifies an open resource within a process, rather than managing user accounts.

B - Incorrect: A descriptor is a handle used to access a resource, not a general hardware-management component.

C - Incorrect: Descriptors support resource I/O and do not manage application lifecycles or software installations.

D - Correct: An open-file or socket reference is the resource-handle purpose of a file descriptor.

Primary note proof

Study-note section OS36

> A file has content and metadata such as name, size, owner, permissions and timestamps. open obtains an open-file reference, read/write transfer bytes, seek changes an offset, close releases a reference, truncate changes size and unlink removes a name. A file descriptor is a process-local integer handle; the associated open-file state contains the current offset and access mode.

Study-note section OS36

> Sockets also support ordinary read/write, alongside socket/connect/send operations, so read is not uniquely “not a network call.” FAT32, NTFS and ext4 are filesystem formats; HTTP is an application communication protocol.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 48 | lines 5-8

13.1 File
A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks. In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the file's creator and user.

Operating System module | PDF page 11 | lines 3-8

- Program needs to read a file or write a file.
- The operating system gives the permission to the program for operation on file.
- Permission varies from read-only, read-write, denied and so on.
- Operating System provides an interface to the user to create/delete files.
- Operating System provides an interface to the user to create/delete directories.
- Operating System provides an interface to create the backup of file system.

Question sources

Operating Systems practice questions · Question 43

Question 82 | Q01454

What is the main function of the command interpreter?

A. to provide the interface between the API and application program

B. to handle the files in the operating system

C. to get and execute the next user-specified command

D. none of the mentioned

Hint

Distinguish command interfaces, privileged core services and the stages of booting.

Answer C: to get and execute the next user-specified command

The shell is a special interpreter that executes OS commands and application programs (PDF p54), taking user commands to kernel functions. C states that command-interpreter loop role.

Why each choice is right or wrong

A - Incorrect: The module places the shell between user commands and kernel, not API and application.

B - Incorrect: File handling is one possible command activity, not the interpreter's general function.

C - Correct: Getting and executing user commands matches p54.

D - Incorrect: C already matches the supplied definition.

Primary note proof

Study-note section OS03

The kernel implements core resource management, protection and system-call handling. A shell reads/interprets user commands; a GUI is another interface. Browsers, office tools and editors are applications.

Study-note section OS04

A user-facing API/library may wrap that request; not every library function enters the kernel.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 54 | lines 18-20

commands of the operating system. It can be used to do various types of operations, call application programs etc.
Security - Linux provides user security using authentication features like password

Operating System module | PDF page 54 | lines 24-30

Hardware layer - Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc.).
Kernel - Core component of Operating System, interacts directly with hardware, provides low level services to upper layer components.
Shell - An interface to kernel, hiding complexity of kernel's functions from users. Takes commands from user and executes kernel's functions.
Utilities - Utility programs giving user most of the functionalities of an operating systems.

Question sources

Operating Systems practice questions · Question 2

Question 83 | Q01458

Which failures can OS software report or handle while it continues executing on powered hardware? C means complete power loss with no backup supply or advance notification. Select all that apply.

A. lack of paper in printer

B. connection failure in the network

C. power failure

D. all of the mentioned

Hint

Separate device hardware, device-specific software and general event or transfer mechanisms.

Answer A, B: lack of paper in printer; connection failure in the network

OS I/O services can report or respond to a printer running out of paper and to a network connection failure. Complete unannounced loss of power without backup prevents the processor from continuing to run the OS, so software alone cannot handle it during continued execution. The aggregate option includes that unsupported case. Under the explicit power assumptions, the printer and network choices are selected.

Why each choice is right or wrong

A - Correct: The OS can report or handle an out-of-paper printer condition while running.

B - Correct: The OS can report or handle a network connection failure while running.

C - Incorrect: Complete unannounced power loss without backup prevents continued software execution.

D - Incorrect: All includes the excluded total-power-loss case and is therefore incorrect.

Primary note proof

Study-note section OS34

> Printer-paper and network errors can be handled/reported, but continued operation during total power loss requires hardware support, not OS software alone.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 47 | lines 17-18

> Error Handling - An operating system that uses protected memory can guard against many kinds of hardware and application errors.

Operating System module | PDF page 45 | lines 10-22

> The CPU hardware uses an interrupt request line wire which helps CPU to sense after executing every instruction. When the CPU checks that a controller has put a signal on the interrupt request line, the CPU saves a state, such as the current value of the instruction pointer, and jumps to the interrupt handler routine at a fixed address. The interrupt handler part determines the cause of the interrupt performs the necessary processing and executes an interrupt instruction to return the CPU to its execution state.
> The basic mechanism of interrupt enables the CPU to respond to an asynchronous event, such as when a device controller becomes ready for service. Most CPUs have two interrupt request lines.
> • non-maskable interrupt - Such kind of interrupts are reserved for events like unrecoverable memory errors.
> • maskable interrupt - Such interrupts can be switched off by the CPU before the execution of critical instructions that must not be interrupted.

Question sources

Operating Systems practice questions · Question 6

Question 84 | Q01459

Who among the following is credited as a co-creator of the UNIX operating system at Bell Labs, together with Ken Thompson?

A. Charles Babbage

B. Alan Turing

C. Tim Berners-Lee

D. Dennis Ritchie

Hint

Compare the stated historical clues and distinguish origin, characteristic era and technical capability.

Answer D: Dennis Ritchie

Dennis Ritchie is credited alongside Ken Thompson in Bell Labs' account of UNIX's creation. The other listed people are not that co-creator.

Why each choice is right or wrong

A - Incorrect: Babbage is not the UNIX co-creator alongside Thompson identified by Bell Labs.

B - Incorrect: Turing is not the UNIX co-creator alongside Thompson identified by Bell Labs.

C - Incorrect: Berners-Lee is not the UNIX co-creator alongside Thompson identified by Bell Labs.

D - Correct: Dennis Ritchie is the Bell Labs co-creator associated with Ken Thompson.

Primary note proof

Study-note section OS-S01

> UNIX development began at Bell Labs in 1969, led by Ken Thompson and Dennis Ritchie with other collaborators. Ritchie is therefore a co-creator, not the sole inventor of the first operating system.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 53 | lines 5-7

> Linux is one of popular version of UNIX operating System. It is open source as its source code is freely available. It is free to use. Linux was designed considering UNIX compatibility. Its functionality list is quite similar to that of UNIX.

Question sources

Operating Systems practice questions · Question 7

Question 85 | Q01460

Which operating system was developed by Microsoft as a successor to MS-DOS?

A. macOS

B. Windows

C. Linux

D. Solaris

Hint

Use the stated historical or platform scope and distinguish origin from technical capability.

Answer B: Windows.

OS40 explicitly identifies Windows as the Microsoft family commonly described as succeeding the MS-DOS desktop era. That directly matches the stem's company and predecessor clues. Linux is separately attributed to Linus Torvalds, Solaris to Sun, and macOS is a different proprietary OS product. No additional market-history claim is needed to resolve the listed alternatives.

Why each choice is right or wrong

A - Incorrect: macOS is not the Microsoft family identified as succeeding the MS-DOS desktop era.

B - Correct: The note explicitly identifies Windows as Microsoft's successor family to the MS-DOS desktop era.

C - Incorrect: Linux is separately attributed to Linus Torvalds, not the Microsoft MS-DOS desktop succession.

D - Incorrect: Solaris originated at Sun rather than being Microsoft's MS-DOS successor family.

Primary note proof

Study-note section OS40

> Windows is the Microsoft family commonly described as succeeding the MS-DOS desktop era; Solaris originated at Sun.

Study-note section OS40

> Linux is a Unix-compatible kernel written from scratch by Linus Torvalds; a distribution combines it with userland, libraries and tools.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 8 | lines 21-26

> Network Operating System runs on a server and provides server the capability to manage data, users, groups, security, applications, and other networking functions. The primary purpose of the network operating system is to allow shared file and printer access among multiple computers in a network, typically a local area network (LAN), a private network or

to other networks. Examples of network operating systems are Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD.

Question sources

Operating Systems practice questions · Question 8

Question 86 | Q01461

Which listed operating-system family originated at Bell Labs in 1969 and later provided multiuser time-sharing with asynchronous processes? This asks for that system's identity, not the first-ever invention of multitasking.

A. MS-DOS

B. macOS

C. Windows

D. UNIX

Hint

Compare the stated historical clues and distinguish origin, characteristic era and technical capability.

Answer D: UNIX

UNIX began at Bell Labs in 1969. Ritchie and Thompson's primary paper describes the later system as multiuser and interactive, with asynchronous processes. These historical identifying facts come from the supplementary sources. OS40 supplies the related caution that UNIX was not the first system ever to multitask. The revised question preserves the history and multitasking focus while dropping that false priority claim.

Why each choice is right or wrong

A - Incorrect: MS-DOS is not the Bell Labs system originating in 1969.

B - Incorrect: macOS is not the original Bell Labs system identified by that history.

C - Incorrect: Windows is not the Bell Labs system described by Ritchie and Thompson.

D - Correct: UNIX matches the Bell Labs origin and the later multiuser time-sharing description.

Primary note proof

Study-note section OS-S01

UNIX development began at Bell Labs in 1969, led by Ken Thompson and Dennis Ritchie with other collaborators. Ritchie is therefore a co-creator, not the sole inventor of the first operating system.

Study-note section OS-S01

The later UNIX time-sharing system described by Ritchie and Thompson is multiuser and interactive and supports asynchronous processes. Its later versions incorporated multiprogramming. This history does not make UNIX the first system ever to multitask.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 13 | lines 21-25

4.3 Multiprogramming
When two or more programs are residing in memory at the same time, then sharing the processor is referred to the multiprogramming. Multiprogramming assumes a single shared processor. Multiprogramming increases CPU utilization by organizing jobs so that the CPU always has one to execute.

Operating System module | PDF page 7 | lines 20-33

Time sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time. Time-sharing or multitasking is a logical extension of multiprogramming. Processor's time which is shared among multiple users simultaneously is termed as time-sharing. The main difference between Multi-programmed Batch Systems and Time-Sharing Systems is that in case of multi-programmed batch systems, objective is to maximize processor use, whereas in Time-Sharing Systems objective is to minimize response time.
Multiple jobs are executed by the CPU by switching between them, but the switches occur so frequently. Thus, the user can receive an immediate response. For example, in a transaction processing, processor execute each user program in a short burst or quantum of computation. That is if n users are present, each user can get time quantum. When the user submits the command, the response time is in few seconds at most.
Operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time. Computer systems that were designed primarily as batch systems

Question sources

Operating Systems practice questions · Question 9

Question 87 | Q01462

Which operating system was developed by Linus Torvalds and is based on the UNIX operating system?

A. Linux

B. Windows

C. macOS

D. Solaris

Hint

Use the stated historical or platform scope and distinguish origin from technical capability.

Answer A: Linux, Unix-compatible and written from scratch.

OS40 attributes Linux to Linus Torvalds and calls it a Unix-compatible kernel written from scratch. The developer clue therefore uniquely identifies Linux. The phrase based on UNIX must be read as compatible/inspired in interface and design, rather than asserting copied UNIX source. The other named families do not match the explicit creator attribution given here.

Why each choice is right or wrong

A - Correct: Linux matches Linus Torvalds and Unix compatibility; the note explicitly says it was written from scratch.

B - Incorrect: Windows is the Microsoft family, not the kernel explicitly attributed here to Linus Torvalds.

C - Incorrect: macOS does not match the note's explicit Torvalds attribution for the Unix-compatible kernel.

D - Incorrect: Solaris originated at Sun rather than being the kernel developed by Linus Torvalds.

Primary note proof

Study-note section OS40

> Linux is a Unix-compatible kernel written from scratch by Linus Torvalds; a distribution combines it with userland, libraries and tools.

Study-note section OS40

> Windows is the Microsoft family commonly described as succeeding the MS-DOS desktop era; Solaris originated at Sun.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 53 | lines 5-7

> Linux is one of popular version of UNIX operating System. It is open source as its source code is freely available. It is free to use. Linux was designed considering UNIX compatibility. Its functionality list is quite similar to that of UNIX.

Question sources

Operating Systems practice questions · Question 10

Question 88 | Q01463

Which operating system was developed by Sun Microsystems and is known for its scalability and reliability?

A. Linux

B. Windows

C. macOS

D. Solaris

Hint

Use the stated historical or platform scope and distinguish origin from technical capability.

Answer D: Solaris.

OS40 directly states that Solaris originated at Sun. That developer clue resolves the choices even though the note does not quantify Solaris's reliability or scalability relative to other systems. The answer relies on the explicit origin, not on assuming those broad positive attributes uniquely distinguish one OS. Linux and Windows have separately identified developer lineages in the same section.

Why each choice is right or wrong

A - Incorrect: Linux is attributed to Linus Torvalds rather than being the Sun-originated OS requested here.

B - Incorrect: Windows is identified as the Microsoft family, not the Sun-originated product.

C - Incorrect: macOS does not match the explicit Sun-origin attribution provided by the complete note.

D - Correct: Solaris is explicitly stated to have originated at Sun, matching the decisive developer clue.

Primary note proof

Study-note section OS40

> Windows is the Microsoft family commonly described as succeeding the MS-DOS desktop era; Solaris originated at Sun.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 29 | lines 4-16

> In this case, thread management done by the Kernel. There is no thread management code in the application area. Kernel threads are supported directly by the operating system. Any application can be programmed to be multithreaded. All of the threads within an application are supported within a single process.
> The Kernel maintains context information for the process as a whole and for individuals' threads within the process. Scheduling by the Kernel is done on a thread basis. The Kernel

performs thread creation, scheduling and management in Kernel space. Kernel threads are generally slower to create and manage than the user threads.
Advantages
• Kernel can simultaneously schedule multiple threads from the same process on multiple processes.
• If one thread in a process is blocked, the Kernel can schedule another thread of the same process.

Question sources

Operating Systems practice questions · Question 11

Question 89 | Q01464

Which listed mobile operating-system platform uses Linux as its base kernel layer and has an open-source base called AOSP?

A. iOS

B. Android

C. Windows Phone

D. BlackBerry OS

Hint

Use the stated historical or platform scope and distinguish origin from technical capability.

Answer B: Android

Android uses Linux at its base kernel layer, with framework and application layers above it. Its open-source base is AOSP, although shipped devices may contain proprietary services or applications. Those architectural facts identify Android without needing an unspecified population or a dated market-share claim. The other listed mobile systems are not the platform identified by this particular combination of Linux and AOSP.

Why each choice is right or wrong

A - Incorrect: iOS is not the platform identified by the Linux/AOSP combination.

B - Correct: Android has the Linux base layer and the AOSP open-source base.

C - Incorrect: Windows Phone is not the Linux/AOSP platform described.

D - Incorrect: BlackBerry OS is not the Linux/AOSP platform described.

Primary note proof

Study-note section OS40

Android’s base kernel layer is Linux; its framework and applications are higher layers. AOSP is open source, while a shipped device can include proprietary applications/services.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 8 | lines 21-26

> Network Operating System runs on a server and provides server the capability to manage data, users, groups, security, applications, and other networking functions. The primary purpose of the network operating system is to allow shared file and printer access among multiple computers in a network, typically a local area network (LAN), a private network or to other networks. Examples of network operating systems are Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD.

Question sources

Operating Systems practice questions · Question 12

Question 90 | Q01471

Using the first four generations described in NYU's OS-history notes (vacuum tubes; transistors; integrated circuits; personal computers), which generation is associated with resident uniprogrammed batch operating systems, following the manual/no-resident-OS era?

A. First Generation

B. Second Generation

C. Third Generation

D. Fourth Generation

Hint

Compare the stated historical clues and distinguish origin, characteristic era and technical capability.

Answer B: Second Generation

The first four generations in NYU's historical taxonomy associates second-generation transistor machines with uniprogrammed batch systems whose OS remains resident between jobs. The first era has no separate resident OS; the third emphasizes ICs and multiprogramming, and the fourth personal computers. OS02 explains batch operation, while the generation mapping is supplied by the supplementary university notes. This is a scoped teaching convention rather than a universal first-OS claim.

Why each choice is right or wrong

A - Incorrect: The first era is the manual/no-separate-resident-OS era in this taxonomy.

B - Correct: The second era pairs transistor machines with resident uniprogrammed batch systems.

C - Incorrect: The third era emphasizes ICs and multiprogramming rather than the initial resident batch stage.

D - Incorrect: The fourth era emphasizes personal computers and follows the batch-monitor era.

Primary note proof

Study-note section OS-S01

> In the first four generations of the cited NYU teaching taxonomy, the first generation uses vacuum tubes and manual single-user operation without a separate resident OS. The second generation pairs transistors with uniprogrammed batch systems; the resident OS starts successive jobs without being reloaded for every job.

Study-note section OS-S01

> The third generation pairs integrated circuits with multiprogramming. The fourth emphasizes personal computers. These are characteristic-era associations, not claims that each feature existed exclusively in one era.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 7 | lines 20-33

> Time sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time. Time-sharing or multitasking is a logical extension of multiprogramming. Processor's time which is shared among multiple users simultaneously is termed as time-sharing. The main difference between Multi-programmed Batch Systems and Time-Sharing Systems is that in case of multi-programmed batch systems, objective is to maximize processor use, whereas in Time-Sharing Systems objective is to minimize response time.
> Multiple jobs are executed by the CPU by switching between them, but the switches occur so frequently. Thus, the user can receive an immediate response. For example, in a transaction processing, processor execute each user program in a short burst or quantum of computation. That is if n users are present, each user can get time quantum. When the user submits the command, the response time is in few seconds at most.
> Operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time. Computer systems that were designed primarily as batch systems

Operating System module | PDF page 7 | lines 5-7

> operating systems are there from the very first computer generation. Operating systems keep evolving over the period of time. Following are few of the important types of operating system which are most commonly used.

Question sources

Operating Systems practice questions · Question 19

Question 91 | Q01475

What is the primary function of a kernel in an operating system?

A. Provide a user-friendly interface

B. Manage system resources and provide core services

C. Run user applications

D. Facilitate communication with external devices

Hint

Distinguish command interfaces, privileged core services and the stages of booting.

Answer B: Manage system resources and provide core services

The kernel is the core, responsible for major activities and direct hardware interaction (PDF p53), providing services with privileged access. B summarizes this rather than a particular interface or device subset.

Why each choice is right or wrong

A - Incorrect: The shell supplies the user-command interface (p54).

B - Correct: Core services and resource access match p53.

C - Incorrect: The kernel runs processes, but application execution alone omits its wider central role.

D - Incorrect: External-device communication is one hardware activity, not the entire primary role.

Primary note proof

Study-note section OS03

> The kernel implements core resource management, protection and system-call handling. A shell reads/interprets user commands; a GUI is another interface. Browsers, office tools and editors are applications.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 53 | lines 11-15

> operating system. It is consisting of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.
> System Library - System libraries are special functions or programs using which application programs or system utilities accesses Kernel's features. These libraries

Operating System module | PDF page 53 | lines 21-29

> Kernel component code executes in a special privileged mode called kernel mode with full access to all resources of the computer. This code represents a single process, executes in single

address space and do not require any context switch and hence is very efficient and fast.
Kernel runs each process and provides system services to processes, provides protected access to hardware to processes.
Support code which is not required to run in kernel mode is in System Library. User programs and other system programs works in User Mode which has no access to system hardware and kernel code. User programs/ utilities use System libraries to access Kernel functions to get system's low level tasks.

Operating System module | PDF page 54 | lines 18-20

commands of the operating system. It can be used to do various types of operations, call application programs etc.
Security - Linux provides user security using authentication features like password

Question sources

Operating Systems practice questions · Question 23

Question 92 | Q01476

Which component of an operating system is responsible for memory management and process scheduling?

A. Kernel

B. User program

C. Compiler

D. File system

Hint

Distinguish command interfaces, privileged core services and the stages of booting.

Answer A: Kernel

Kernel is the major-activity core (PDF p53); memory management and ready-process selection are OS functions (pp31,20). Under the module's architecture, A is the component responsible for these core services.

Why each choice is right or wrong

A - Correct: Kernel provides the core resource services (p53).

B - Incorrect: User programs request services rather than being the core manager (p53).

C - Incorrect: Compilation is not the ready-process or memory manager described.

D - Incorrect: File organization does not substitute for memory/CPU management.

Primary note proof

Study-note section OS03

The kernel implements core resource management, protection and system-call handling.

Study-note section OS01

An OS allocates CPU time, memory, storage, devices and communication resources, and provides services to programs.

Study-note section OS06

Long-term scheduling admits jobs; medium-term scheduling suspends/swaps them; short-term scheduling selects ready work.

Study-note section OS-S05

A compiler translates source code toward an executable representation. For example, the conventional GCC toolchain distinguishes preprocessing, compilation proper, assembly and linking. A compiler uses operating-system services as a development tool; it is not the runtime kernel component that selects ready processes, manages memory or handles protected service requests.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 53 | lines 11-15

operating system. It is consisting of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.
System Library - System libraries are special functions or programs using which application programs or system utilities accesses Kernel's features. These libraries

Operating System module | PDF page 31 | lines 24-31

Memory management is the functionality of an operating system which handles or manages primary memory. Memory management keeps track of each and every memory location either it is allocated to some process or it is free. It checks how much memory is to be allocated to processes. It decides which process will get memory at what time. It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status.
Memory management provides protection by using two registers, a base register and a limit register. The base register holds the smallest legal physical memory address and the limit register specifies the size of the range. For example, if the base register holds 300000 and the

Operating System module | PDF page 20 | lines 15-21

It is also called CPU scheduler. Main objective is increasing system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. CPU scheduler selects process among the processes that are ready to execute and allocates CPU to one of them.
Short term scheduler also known as dispatcher, execute most frequently and makes the fine-grained decision of which process to execute next. Short term scheduler is faster than long term scheduler.

Question sources

Operating Systems practice questions · Question 24

Question 93 | Q01477

In the module’s conventional architecture, where device drivers are kernel components and ordinary applications run with restricted privilege, which item normally executes in kernel mode?

A. User programs

B. Device drivers

C. Text editors

D. Web browsers

Hint

Distinguish command interfaces, privileged core services and the stages of booting.

Answer B: Device drivers

Kernel components perform privileged operating-system work. In the stated architecture, device drivers are such components, so they execute in kernel mode. User programs, text editors and web browsers are ordinary applications and execute in restricted user mode, requesting protected services when needed. User-space driver designs exist, which is why the architecture is stated rather than treating every driver implementation as privileged.

Why each choice is right or wrong

A - Incorrect: Ordinary user programs execute in restricted user mode.

B - Correct: The specified drivers are kernel components and therefore execute with kernel privilege.

C - Incorrect: A text editor is an ordinary application, so it runs in user mode under this architecture.

D - Incorrect: A web browser is an ordinary application and does not acquire kernel privilege merely by using OS services.

Primary note proof

Study-note section OS03

> Browsers, office tools and editors are applications. Device drivers are commonly kernel components, although user-space drivers also exist.

Study-note section OS-S02

> Being scheduled or managed by the kernel does not give application thread code kernel privilege. Ordinary applications and supporting utilities execute in restricted user mode and request protected services through libraries and system calls. Kernel component code executes in privileged kernel mode and performs permitted low-level hardware operations.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 45 | lines 28-33

> Application I/O Interface represents the structuring techniques and interfaces for the operating system to enable I/O devices to be treated in a standard, uniform way. The actual differences lies kernel level modules called device drivers which are custom tailored to corresponding devices but show one of the standard interfaces to applications. The purpose of the device-driver layer is to hide the differences among device controllers from the I/O subsystem of the kernel, such as the I/O system calls. Following are the characteristics of I/O

Operating System module | PDF page 53 | lines 21-29

> Kernel component code executes in a special privileged mode called kernel mode with full access to all resources of the computer. This code represents a single process, executes in single address space and do not require any context switch and hence is very efficient and fast. Kernel runs each process and provides system services to processes, provides protected access to hardware to processes.
> Support code which is not required to run in kernel mode is in System Library. User programs and other system programs works in User Mode which has no access to system hardware and kernel code. User programs/ utilities use System libraries to access Kernel functions to get system's low level tasks.

Question sources

Operating Systems practice questions · Question 25

Question 94 | Q01478

In the operating system’s user/kernel privilege model, which standard term names the restricted level used by ordinary application code? Classify execution privilege, rather than processor addressing modes.

A. Kernel mode

B. Supervisor mode

C. User mode

D. Protected mode

Hint

Distinguish where threads are managed from the privilege of the code they execute.

Answer C: User mode

Ordinary application code runs at the restricted user level and obtains protected services through libraries and system calls. Kernel mode is the privileged level; supervisor mode is commonly another name for that privileged level. Protected mode is not the name of the restricted application privilege level in the stated user/kernel classification. The question therefore selects User mode without making a claim about every processor’s architectural modes.

Why each choice is right or wrong

A - Incorrect: Kernel mode permits privileged OS operations and is not the restricted application level.

B - Incorrect: Supervisor mode denotes the privileged context in this model, rather than ordinary application privilege.

C - Correct: User mode is the restricted application level in the user/kernel privilege model.

D - Incorrect: The requested restricted privilege level is user mode; an architectural mode label does not name that level in this classification.

Primary note proof

Study-note section OS-S02

> Being scheduled or managed by the kernel does not give application thread code kernel privilege. Ordinary applications and supporting utilities execute in restricted user mode and request protected services through libraries and system calls. Kernel component code executes in privileged kernel mode and performs permitted low-level hardware operations.

Study-note section OS03

> Kernel mode has privileged operations; supervisor mode often means the same privileged mode, so offering both as exclusive answers is ambiguous.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 53 | lines 21-29

> Kernel component code executes in a special privileged mode called kernel mode with full access to all resources of the computer. This code represents a single process, executes in single address space and do not require any context switch and hence is very efficient and fast.
> Kernel runs each process and provides system services to processes, provides protected access to hardware to processes.
> Support code which is not required to run in kernel mode is in System Library. User programs and other system programs works in User Mode which has no access to system hardware and kernel code. User programs/ utilities use System libraries to access Kernel functions to get system's low level tasks.

Question sources

Operating Systems practice questions · Question 26

Question 95 | Q01479

In the teaching module’s architecture, which broad core component provides protected file-system operations for applications?

A. Kernel

B. User program

C. Device driver

D. Compiler

Hint

Distinguish command interfaces, privileged core services and the stages of booting.

Answer A: Kernel

The kernel provides core protected services and handles system calls, including the operating system’s file operations in this architecture. Applications request those services. A device driver translates requests into device-specific operations beneath the file-service layer, while a compiler translates program source. These supporting roles do not make either the broad core provider of protected file-system operations.

Why each choice is right or wrong

A - Correct: The kernel provides the protected OS service and system-call layer in the stated architecture.

B - Incorrect: Applications request file services rather than being the protected core service provider.

C - Incorrect: Drivers perform device-specific operations beneath the broader file-service layer.

D - Incorrect: A compiler translates program source; it is not the runtime protected file-service provider.

Primary note proof

Study-note section OS03

> The kernel implements core resource management, protection and system-call handling.

Study-note section OS36

> A file has content and metadata such as name, size, owner, permissions and timestamps. open obtains an open-file reference, read/write transfer bytes, seek changes an offset, close releases a reference, truncate changes size and unlink removes a name.

Study-note section OS34

> A driver converts OS requests into device-specific operations.

Study-note section OS-S05

> A compiler translates source code toward an executable representation. For example, the conventional GCC toolchain distinguishes preprocessing, compilation proper, assembly and

linking. A compiler uses operating-system services as a development tool; it is not the runtime kernel component that selects ready processes, manages memory or handles protected service requests.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 53 | lines 11-15

operating system. It is consisting of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.
System Library - System libraries are special functions or programs using which application programs or system utilities accesses Kernel's features. These libraries

Operating System module | PDF page 6 | lines 19-23

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions. Operating System does the following activities for file management.
Keeps track of information, location, uses, status etc. The collective facilities are often known as file system.

Operating System module | PDF page 45 | lines 28-33

Application I/O Interface represents the structuring techniques and interfaces for the operating system to enable I/O devices to be treated in a standard, uniform way. The actual differences lies kernel level modules called device drivers which are custom tailored to corresponding devices but show one of the standard interfaces to applications. The purpose of the device-driver layer is to hide the differences among device controllers from the I/O subsystem of the kernel, such as the I/O system calls. Following are the characteristics of I/O

Question sources

Operating Systems practice questions · Question 27

Question 96 | Q01480

During a conventional disk-based boot, the bootloader reads the kernel image from disk. Into which main execution-memory region is it loaded before control is transferred to it?

A. RAM

B. Hard disk

C. CPU cache

D. ROM

Hint

Distinguish command interfaces, privileged core services and the stages of booting.

Answer A: RAM

In the stated boot sequence, firmware and bootloader stages run before control passes to the kernel loaded into RAM. The disk supplies the stored image, while RAM is its main execution-memory destination. CPU cache can retain copies of memory contents for faster reuse, but that is distinct from the image's main-memory loading location. ROM is not the RAM destination specified by this conventional boot model.

Why each choice is right or wrong

A - Correct: RAM is the main execution-memory destination for the loaded kernel image in the stated boot sequence.

B - Incorrect: The hard disk is the source of the stored image here, not its main-memory loading destination.

C - Incorrect: Cache retains copies for reuse; it is not the bootloader's main-memory image region.

D - Incorrect: ROM does not identify the writable RAM destination of the kernel image in this stated disk-boot model.

Primary note proof

Study-note section OS03

> Firmware/bootloader stages execute before transferring control to a kernel loaded into RAM.

Study-note section OS35

> Caching retains copies to accelerate reuse.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 5 | lines 26-36

> Memory management refers to management of Primary Memory or Main Memory. Main memory is a large array of words or bytes where each word or byte has its own address. Main memory provides a fast storage that can be access directly by the CPU. So, for a program to be executed, must in the main memory.
> Activities Operating System for memory management.
> Keeps tracks of primary memory (i.e. what part of it are in use by whom, what part are not in use.)
> In multiprogramming, OS decides which process will get memory when and how much.
> Allocates the memory when the process requests it to do so.
> De-allocates the memory when the process no longer needs it or has been terminated.

Operating System module | PDF page 31 | lines 24-31

> Memory management is the functionality of an operating system which handles or manages primary memory. Memory management keeps track of each and every memory location either it is allocated to some process or it is free. It checks how much memory is to be allocated to processes. It decides which process will get memory at what time. It tracks whenever some

memory gets freed or unallocated and correspondingly it updates the status.
Memory management provides protection by using two registers, a base register and a limit register. The base register holds the smallest legal physical memory address and the limit register specifies the size of the range. For example, if the base register holds 300000 and the

Operating System module | PDF page 33 | lines 25-28

9.5 Memory Allocation
Main memory usually has two partitions
• Low Memory -- Operating system resides in this memory.
• High Memory -- User processes then held in high memory.

Question sources

Operating Systems practice questions · Question 28

Question 97 | Q01481

In a conventional monolithic OS, which listed utilities can have user-space program components that invoke privileged OS services, even if the complete product also includes privileged helpers? Treat “device driver” and “interrupt handler” here as kernel-resident components. Select all that apply.

A. Device driver

B. Antivirus software

C. Disk partition manager

D. Interrupt handler

Hint

Distinguish command interfaces, privileged core services and the stages of booting.

Answer B, C: Antivirus software; Disk partition manager

Antivirus and disk-management products can combine ordinary user processes with privileged helper components. Their user-space components count as user programs even though some work requires kernel services. The stem explicitly confines the driver and interrupt-handler options to kernel-resident components, avoiding architectures with user-space drivers. Thus antivirus software and the partition-manager utility are both valid examples under the stated component-level interpretation.

Why each choice is right or wrong

A - Incorrect: The driver is explicitly kernel-resident in this question.

B - Correct: Antivirus software can have a user-process component alongside privileged helpers.

C - Correct: A disk partition manager can have a user-process component invoking privileged services.

D - Incorrect: The interrupt handler is explicitly kernel-resident in this question.

Primary note proof

Study-note section OS03

> Utilities such as antivirus and disk tools can combine user processes with privileged components.

Study-note section OS-S02

> Being scheduled or managed by the kernel does not give application thread code kernel privilege. Ordinary applications and supporting utilities execute in restricted user mode and request protected services through libraries and system calls. Kernel component code executes in privileged kernel mode and performs permitted low-level hardware operations.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 53 | lines 21-29

> Kernel component code executes in a special privileged mode called kernel mode with full access to all resources of the computer. This code represents a single process, executes in single address space and do not require any context switch and hence is very efficient and fast.
> Kernel runs each process and provides system services to processes, provides protected access to hardware to processes.
> Support code which is not required to run in kernel mode is in System Library. User programs and other system programs works in User Mode which has no access to system hardware and kernel code. User programs/ utilities use System libraries to access Kernel functions to get system's low level tasks.

Operating System module | PDF page 47 | lines 19-26

> 12.5 Device driver
> Device driver is a program or routine developed for an I/O device. A device driver implements I/O operations or behaviors on a specific class of devices. For example a system supports one or a number of multiple brands of terminals, all slightly different terminals may have a single terminal driver. In the layered structure of I/O system, device driver lies between interrupt handler and device independent I/O software. The job of a device driver is following.
> To accept request from the device independent software above it.
> To see to it that the request is executed.

Operating System module | PDF page 45 | lines 10-22

> The CPU hardware uses an interrupt request line wire which helps CPU to sense after executing every instruction. When the CPU checks that a controller has put a signal on the interrupt request line, the CPU saves a state, such as the current value of the instruction pointer, and jumps to the interrupt handler routine at a fixed address. The interrupt handler part determines the cause of the interrupt performs the necessary processing and executes an interrupt instruction to return the CPU to its execution state.
> The basic mechanism of interrupt enables the CPU to respond to an asynchronous event, such as when a device controller becomes ready for service. Most CPUs have two interrupt request lines.

• non-maskable interrupt - Such kind of interrupts are reserved for events like unrecoverable memory errors.
• maskable interrupt - Such interrupts can be switched off by the CPU before the execution of critical instructions that must not be interrupted.

Question sources

Operating Systems practice questions · Question 29

Question 98 | Q01482

In the module's operating-system architecture, which broad core component performs privileged interaction with underlying hardware?

A. Kernel

B. User program

C. Compiler

D. File system

Hint

Distinguish where threads are managed from the privilege of the code they execute.

Answer A: Kernel

The kernel performs privileged low-level hardware operations and supplies protected services. Ordinary applications use those services from user mode. A compiler translates source code, and the file system provides a particular family of storage services rather than naming the broad hardware-facing OS core. The question asks for that broad core component, not a particular device-specific driver.

Why each choice is right or wrong

A - Correct: The kernel is the privileged OS core that performs permitted low-level hardware operations.

B - Incorrect: Ordinary user programs request protected services rather than directly obtaining kernel privilege.

C - Incorrect: A compiler performs source translation, a development-tool role rather than privileged OS-core hardware management.

D - Incorrect: A file system supplies specific storage services, not the broad hardware-facing OS core requested.

Primary note proof

Study-note section OS-S02

Being scheduled or managed by the kernel does not give application thread code kernel privilege. Ordinary applications and supporting utilities execute in restricted user mode and

request protected services through libraries and system calls. Kernel component code executes in privileged kernel mode and performs permitted low-level hardware operations.

Study-note section OS-S05

A compiler translates source code toward an executable representation. For example, the conventional GCC toolchain distinguishes preprocessing, compilation proper, assembly and linking. A compiler uses operating-system services as a development tool; it is not the runtime kernel component that selects ready processes, manages memory or handles protected service requests.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 53 | lines 11-15

operating system. It is consisting of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.
System Library - System libraries are special functions or programs using which application programs or system utilities accesses Kernel's features. These libraries

Operating System module | PDF page 53 | lines 21-29

Kernel component code executes in a special privileged mode called kernel mode with full access to all resources of the computer. This code represents a single process, executes in single address space and do not require any context switch and hence is very efficient and fast.
Kernel runs each process and provides system services to processes, provides protected access to hardware to processes.
Support code which is not required to run in kernel mode is in System Library. User programs and other system programs works in User Mode which has no access to system hardware and kernel code. User programs/ utilities use System libraries to access Kernel functions to get system's low level tasks.

Question sources

Operating Systems practice questions · Question 30

Question 99 | Q01483

In a two-level privilege model where “kernel mode” and “supervisor mode” are synonymous names for the privileged execution level, which listed names denote that level? “Protected mode” denotes a processor operating mode that can include restricted user execution. Select all that apply.

A. Kernel mode

B. User mode

C. Supervisor mode

D. Protected mode

Hint

Distinguish command interfaces, privileged core services and the stages of booting.

Answer A, C: Kernel mode; Supervisor mode

The note states that kernel mode permits privileged operations and that supervisor mode commonly denotes the same privileged mode. The stated model makes those names explicitly synonymous. User mode is restricted. Protected mode is defined here as a broader processor mode that can still run user code, so its name alone does not identify the privileged execution level. Both kernel and supervisor are selected.

Why each choice is right or wrong

A - Correct: Kernel mode denotes the privileged execution level.

B - Incorrect: User mode is the restricted level.

C - Correct: Supervisor mode is explicitly synonymous with kernel mode in this model.

D - Incorrect: Protected mode is not a privilege level under the supplied definition.

Primary note proof

Study-note section OS03

> Kernel mode has privileged operations; supervisor mode often means the same privileged mode, so offering both as exclusive answers is ambiguous.

Study-note section OS-S02

> Being scheduled or managed by the kernel does not give application thread code kernel privilege. Ordinary applications and supporting utilities execute in restricted user mode and request protected services through libraries and system calls. Kernel component code executes in privileged kernel mode and performs permitted low-level hardware operations.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 53 | lines 21-29

> Kernel component code executes in a special privileged mode called kernel mode with full access to all resources of the computer. This code represents a single process, executes in single address space and do not require any context switch and hence is very efficient and fast.
> Kernel runs each process and provides system services to processes, provides protected access to hardware to processes.
> Support code which is not required to run in kernel mode is in System Library. User programs and other system programs works in User Mode which has no access to system hardware and kernel code. User programs/ utilities use System libraries to access Kernel functions to get system's low level tasks.

Question sources

Operating Systems practice questions · Question 31

Question 100 | Q01484

Which of the following is responsible for handling system calls made by user programs?

A. Kernel

B. User program

C. Compiler

D. File system

Hint

Distinguish command interfaces, privileged core services and the stages of booting.

Answer A: Kernel

PDF p53 says the kernel provides system services and protected hardware access to processes; libraries let user code reach kernel functions. Thus the handling component is the kernel, not the application caller or compiler.

Why each choice is right or wrong

A - Correct: Kernel provides the requested process services (p53).

B - Incorrect: User program is the requester using libraries (p53).

C - Incorrect: Compiler is not the runtime service handler.

D - Incorrect: File system handles one service category, not all kernel requests.

Primary note proof

Study-note section OS03

> The kernel implements core resource management, protection and system-call handling.

Study-note section OS04

> A system call requests a protected kernel service, such as creating a process, reading a file or opening a socket. A user-facing API/library may wrap that request; not every library function enters the kernel.

Study-note section OS-S05

> A compiler translates source code toward an executable representation. For example, the conventional GCC toolchain distinguishes preprocessing, compilation proper, assembly and linking. A compiler uses operating-system services as a development tool; it is not the runtime kernel component that selects ready processes, manages memory or handles protected service requests.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 53 | lines 21-29

> Kernel component code executes in a special privileged mode called kernel mode with full access to all resources of the computer. This code represents a single process, executes in single address space and do not require any context switch and hence is very efficient and fast.
> Kernel runs each process and provides system services to processes, provides protected access to hardware to processes.
> Support code which is not required to run in kernel mode is in System Library. User programs and other system programs works in User Mode which has no access to system hardware and kernel code. User programs/ utilities use System libraries to access Kernel functions to get system's low level tasks.

Operating System module | PDF page 53 | lines 16-19

> implement most of the functionalities of the operating system and do not requires kernel module's code access rights.
> System Utility - System Utility programs are responsible to do specialized, individual level tasks.

Question sources

Operating Systems practice questions · Question 32

Question 101 | Q01506

A scheduler selects the next process to be executed from one of the following.

A. Ready queue

B. Job queue

C. Device queue

D. None of the choices is correct

Hint

Decide whether the process is eligible, running or waiting, then identify the responsible queue or scheduling role.

Answer A: Ready queue

PDF pp19-20 place processes waiting for CPU in the ready queue and select among ready processes. The question is about the next CPU execution, hence A rather than job admission.

Why each choice is right or wrong

A - Correct: Ready queue contains eligible CPU waiters.

B - Incorrect: Job queue includes all processes/admission work, not just eligible CPU contenders.

C - Incorrect: Device queue contains I/O waiters (p18).

D - Incorrect: Ready queue is explicitly supplied.

Primary note proof

Study-note section OS06

The ready queue contains eligible work; device queues hold blocked requests. The broad textbook job queue contains processes in the system, while an admission queue more narrowly holds jobs awaiting memory/admission. Long-term scheduling admits jobs; medium-term scheduling suspends/swaps them; short-term scheduling selects ready work.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 19 | lines 6-12

A newly arrived process is put in the ready queue. Processes waits in ready queue for allocating the CPU. Once the CPU is assigned to a process, then that process will execute. While executing the process, any one of the following events can occur.
• The process could issue an I/O request and then it would be placed in an I/O queue.
• The process could create new sub process and will wait for its termination.
• The process could be removed forcibly from the CPU, as a result of interrupt and put back in the ready queue.

Operating System module | PDF page 20 | lines 15-21

It is also called CPU scheduler. Main objective is increasing system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. CPU scheduler selects process among the processes that are ready to execute and allocates CPU to one of them.
Short term scheduler also known as dispatcher, execute most frequently and makes the fine-grained decision of which process to execute next. Short term scheduler is faster than long term scheduler.

Question sources

Mock exam 1 · Question 22

Question 102 | Q01507

Under preemptive priority scheduling, which event sends a currently running, unfinished process back to the ready state?

A. Arrival of a higher-priority ready process that preempts it

B. Completion of its final CPU burst

C. Suspension while waiting for an event

D. Issuing a blocking I/O request

Hint

Decide whether the process is eligible, running or waiting, then identify the responsible queue or scheduling role.

Answer A: Arrival of a higher-priority ready process that preempts it

Preemption removes the CPU from a running process that can still execute, returning it to ready. In the stated priority policy, arrival of a higher-priority ready process triggers that handoff. Final completion ends execution, while a blocking I/O request makes the process wait for an event. Suspension while waiting also does not describe ordinary readiness, so only the explicit preemption event fits.

Why each choice is right or wrong

A - Correct: The higher-priority arrival preempts the runnable process and returns it to ready.

B - Incorrect: Final completion transitions to terminated, not ready.

C - Incorrect: Suspended waiting work is not eligible for the ordinary ready queue.

D - Incorrect: Blocking I/O transitions running to waiting.

Primary note proof

Study-note section OS06

> New processes are created/admitted; ready means eligible for CPU but not currently running. Running uses a CPU. Waiting/blocked means unable to proceed until an event; terminated means execution has ended. Running→ready follows time-slice expiry or another preemption. Running→waiting follows blocking I/O. I/O completion normally gives waiting→ready, not a guaranteed immediate running state.

Study-note section OS17

> Priority scheduling chooses the highest-priority ready job. Check whether a smaller or larger numeric value means higher priority. State whether a high-priority arrival preempts.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 19 | lines 6-12

> A newly arrived process is put in the ready queue. Processes waits in ready queue for allocating the CPU. Once the CPU is assigned to a process, then that process will execute. While executing the process, any one of the following events can occur.
> • The process could issue an I/O request and then it would be placed in an I/O queue.
> • The process could create new sub process and will wait for its termination.
> • The process could be removed forcibly from the CPU, as a result of interrupt and put back in the ready queue.

Operating System module | PDF page 24 | lines 10-15

> 7.3 Priority Based Scheduling
> • Each process is assigned a priority. Process with highest priority is to be executed first and so on.
> • Processes with same priority are executed on first come first serve basis.
> • Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Operating System module | PDF page 16 | lines 29-39

1. New The process is being created.
2.
Ready The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run.
3.
Running Process instructions are being executed (i.e. The process that is currently being executed).
4.
Waiting The process is waiting for some event to occur (such as the completion of an I/O operation).
5. Terminated The process has finished execution.

Question sources

Mock exam 1 · Question 23

Question 103 | Q01508

One of the following deadlock handling methods is used if it occurs rarely and is insignificant.

A. Prevention

B. waiting

C. recovery

D. Ignorance

Hint

Separate handling before allocation from actions taken after detection, and consider the cost of checking.

Answer D: Ignorance, the intentional ignoring/ostrich policy.

The ostrich or ignoring policy accepts the risk of a sufficiently rare, insignificant deadlock problem when systematic handling is not worthwhile. That is the intended meaning of Ignorance. Prevention instead constrains requests to break a necessary deadlock condition, and recovery acts after detection. Merely waiting is not the same as deliberately choosing to tolerate the rare problem.

Why each choice is right or wrong

A - Incorrect: Prevention constrains resource requests to remove a necessary deadlock condition; it does not deliberately ignore the problem.

B - Incorrect: Waiting describes a process behavior, not the deliberate policy of accepting rare deadlocks.

C - Incorrect: Recovery takes action after deadlock detection rather than choosing to ignore the rare problem.

D - Correct: Ignorance names the intentional ignoring or ostrich policy in this context.

Primary note proof

Study-note section OS26

> Ignoring a sufficiently rare problem is sometimes called the ostrich policy, but real systems still prevent or detect selected deadlocks. “Deadlock acceptance” can mean intentional ignoring, not necessarily an invalid approach.

Study-note section OS24

> Prevention constrains how requests are made. Share a resource where safe to remove mutual exclusion; request all needed resources before holding any to remove hold-and-wait; permit safe resource withdrawal/restart to attack no-preemption; impose a global acquisition order to remove circular wait.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

> • Loads a program into memory.
> • Executes the program.
> • Handles program's execution.
> • Provides a mechanism for process synchronization.
> • Provides a mechanism for process communication.
> • Provides a mechanism for deadlock handling.

Question sources

Mock exam 1 · Question 24

Question 104 | Q01587

One of the following is false about linked allocation.

A. Is based on sequential file access

B. Use pointer concept

C. No external fragmentation

D. The size of the file should be set in advanced

Hint

Trace how logical file blocks reach physical blocks and account for pointer metadata.

Answer D: The size of the file should be set in advanced

PDF p49 says linked allocation carries links to disk blocks, is effective for sequential access and has no external fragmentation. No advance contiguous size reservation is part of that linked structure; D confuses it with a fixed extent requirement.

Why each choice is right or wrong

A - Incorrect: Sequential access effectiveness is explicitly stated on p49.

B - Incorrect: Links/pointers are explicitly stated on p49.

C - Incorrect: No external fragmentation is explicitly stated on p49.

D - Correct: A chain can link another block without requiring one pre-sized contiguous extent; no advance-size requirement follows from the p49 linked structure.

Primary note proof

Study-note section OS38

> Contiguous allocation uses consecutive blocks; logical block i is start+i and supports direct access, but growth and external fragmentation are problems. Extents allow several contiguous runs. Linked allocation chains blocks and supports growth without reserving a complete contiguous file; locating block i requires traversing preceding links. FAT stores chains in a table; caching that table does not change the chain into a direct arithmetic mapping.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 49 | lines 30-35

> 13.3.2 Linked Allocation
> Each file carries a list of links to disk blocks.
> Directory contains link / pointer to first block of a file.
> No external fragmentation
> Effectively used in sequential access file.
> Inefficient in case of direct access file.

Operating System module | PDF page 48 | lines 5-8

> 13.1 File
> A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks. In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the file’s creator and user.

Question sources

Mock exam 3 · Question 19

Question 105 | Q01588

Consider three specified implementations: an Excel application process with several worker threads, a client-browser process with multiple threads sharing its address space, and a server process using only its single main thread. Which listed process does not use multithreading?

A. Excel application

B. Client browser

C. Server

D. None of the choices is correct

Hint

Compare per-thread execution context with shared process resources and the stated thread mapping.

Answer C: Server

Multithreading means that one process contains multiple threads of execution. The described spreadsheet and browser processes each have several threads sharing their process resources. The specified server implementation has only its main thread and therefore is not multithreaded. This conclusion follows from the supplied designs, not from a universal claim about every version of Excel, every browser, or every server.

Why each choice is right or wrong

A - Incorrect: The specified Excel process has several worker threads and is multithreaded.

B - Incorrect: The specified browser process has multiple threads and is multithreaded.

C - Correct: The specified server has only one thread and is not multithreaded.

D - Incorrect: C is a valid non-multithreaded implementation, so none is incorrect.

Primary note proof

Study-note section OS08

> A thread has its own PC, registers and stack. Threads in a process share code, address space and usually open-file resources. A process can contain many threads; a thread belongs to one process at a time. Threads are not a group of processes and need not execute sequentially.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 27 | lines 6-14

> A thread is a flow of execution through the process code, with its own program counter, system registers and stack. A thread is also called a light weight process. Threads provide a way to improve application performance through parallelism. Threads represent a software approach to improving performance of operating system by reducing the overhead thread is equivalent to a classical process.
> Each thread belongs to exactly one process and no thread can exist outside a process. Each

thread represents a separate flow of control. Threads have been successfully used in implementing network servers and web server. They also provide a suitable foundation for parallel execution of applications on shared memory multiprocessors. Following figure shows

Question sources

Mock exam 3 · Question 20

Question 106 | Q01589

One of the following security techniques is not supported by modern operating systems.

A. Cryptography

B. Authentication

C. Firewall

D. None of the above

Hint

Distinguish identity checks, permission checks and the particular security property at issue.

Answer D: None: modern OS families can support all three techniques.

OS39 states that modern OS families can support cryptography, authentication and firewalls, while support/configuration depends on product. Therefore none of the three is categorically unsupported by modern operating systems. The answer is a capability-category statement, not a guarantee that every installation includes every feature or enables it by default. Each technique has a distinct security role.

Why each choice is right or wrong

A - Incorrect: Cryptography is explicitly a capability that modern OS families can support, though configuration varies.

B - Incorrect: Authentication is explicitly supported as a modern OS capability for verifying identity.

C - Incorrect: Firewalls are explicitly within modern OS support capabilities, even though product and configuration matter.

D - Correct: All three techniques can be supported; none is categorically outside modern operating-system security capabilities.

Primary note proof

Study-note section OS39

Password hashing verifies secrets without storing plaintext; one-time passwords expire after use. Encryption protects confidentiality under correct key use but does not automatically provide integrity. Trojans masquerade as useful software, worms propagate independently and viruses

attach to hosts; denial of service harms availability. Modern OS families can support cryptography, authentication and firewalls, but support/configuration depends on product.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 50 | lines 18-28

Authentication refers to identifying each user of the system and associating the executing programs with those users. It is the responsibility of the Operating System to create a protection system which ensures that a user who is running a particular program is authentic. Operating Systems generally identifies/authenticates users using following three ways:
Username / Password - User need to enter a registered username and password with Operating system to login into the system.
User card/key - User need to punch card in card slot, or enter key generated by key generator in option provided by operating system to login into the system.
User attribute - fingerprint/ eye retina pattern/ signature - User need to pass his/her attribute via designated input device used by operating system to login into the system.

Operating System module | PDF page 54 | lines 5-17

Portable - Portability means software can works on different types of hardware in same way. Linux kernel and application programs support their installation on any kind of hardware platform.
Open Source - Linux source code is freely available and it is community-based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
Multi-User - Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.
Multiprogramming - Linux is a multiprogramming system means multiple applications can run at same time.
Hierarchical File System - Linux provides a standard file structure in which system files/ user files are arranged.
Shell - Linux provides a special interpreter program which can be used to execute

Operating System module | PDF page 54 | lines 5-21

Portable - Portability means software can works on different types of hardware in same way. Linux kernel and application programs support their installation on any kind of hardware platform.
Open Source - Linux source code is freely available and it is community-based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
Multi-User - Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.
Multiprogramming - Linux is a multiprogramming system means multiple applications can run at same time.
Hierarchical File System - Linux provides a standard file structure in which system files/ user files are arranged.

Shell - Linux provides a special interpreter program which can be used to execute commands of the operating system. It can be used to do various types of operations, call application programs etc.
Security - Linux provides user security using authentication features like password protection/ controlled access to specific files/ encryption of data.

Question sources

Mock exam 3 · Question 21

Question 107 | Q01627

Under the basic fixed-page and variable-length-segment memory models, which listed differences are correct? Select all that apply.

A. Paging leads to internal fragmentation whereas segmentation leads external fragmentation

B. The size of block is fixed in the case of paging but variable in the case of segmentation

C. Paging use page table whereas segmentation use segmentation table

D. Paging is visible to user whereas segmentation is invisible to user

Hint

Check the logical region and offset bound before computing its physical location.

Answer A, B, C: Paging leads to internal fragmentation whereas segmentation leads external fragmentation; The size of block is fixed in the case of paging but variable in the case of segmentation; Paging use page table whereas segmentation use segmentation table

Basic paging uses fixed-size pages and page tables, and its final allocated page can contain unused space. Basic segmentation uses variable-length logical units and segment tables, so free holes can cause external fragmentation. Those properties make A, B and C valid contrasts. D reverses the usual abstraction relationship: segments reflect program structure, while ordinary page placement is largely transparent to the application.

Why each choice is right or wrong

A - Correct: Paging can waste space internally; variable-length segmentation can fragment externally.

B - Correct: Pages have a fixed size while segments have variable lengths.

C - Correct: Paging uses page tables and segmentation uses segment tables.

D - Incorrect: Segments reflect program structure; ordinary pages are largely transparent, reversing this option.

Primary note proof

Study-note section OS31

Under the basic models, paging uses fixed units/page tables and can waste space internally; segmentation uses variable logical units/segment tables and can fragment externally. Segments reflect program structure whereas ordinary pages are largely transparent to applications. Apply all three contrasts when comparing paging and segmentation.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 34 | lines 47-56

External fragmentation is avoided by using paging technique. Paging is a technique in which physical memory is broken into blocks of the same size called pages (size is power of 2, between 512 bytes and 8192 bytes). When a process is to be executed, it's corresponding pages are loaded into any available memory frames.
Logical address space of a process can be non-contiguous and a process is allocated physical memory whenever the free memory frame is available. Operating system keeps track of all free frames. Operating system needs n free frames to run a program of size n pages.
Address generated by CPU is divided into
• Page number (p) -- page number is used as an index into a page table which contains base address of each page in physical memory.

Operating System module | PDF page 34 | lines 34-45

External
fragmentation
Total memory space is enough to satisfy a request or to reside a
process in it, but it is not contiguous so it cannot be used.
2
Internal
fragmentation
Memory block assigned to process is bigger. Some portion of memory
is left unused as it cannot be used by another process.
External fragmentation can be reduced by compaction or shuffle memory contents to place all free memory together in one large block. To make compaction feasible, relocation should be dynamic

Operating System module | PDF page 35 | lines 7-12

Segmentation is a technique to break memory into logical pieces where each piece represents a group of related information. For example, data segments or code segment for each process, data segment for operating system and so on. Segmentation can be implemented using or without using paging.
Unlike paging, segment is having varying sizes and thus eliminates internal fragmentation. External fragmentation still exists but to lesser extent.

Operating System module | PDF page 36 | lines 3-7

Address generated by CPU is divided into
• Segment number (s) -- segment number is used as an index into a segment table which contains base address of each segment in physical memory and a limit of segment.

• Segment offset (o) -- segment offset is first checked against limit and then is combined with base address to define the physical memory address.

Question sources

Mock exam 2 · Question 16

Question 108 | Q01628

One of the following IO services is used store data in memory while transferring between devices.

A. Buffering

B. Scheduling

C. Caching

D. Spooling

Hint

Identify whether data is in transit, retained for reuse or queued for a device.

Answer A: Buffering

PDF p47 defines buffering as retaining data while transferred between devices or an application and device. This is A; caching stores reusable copies and spooling queues a noninterleavable output stream.

Why each choice is right or wrong

A - Correct: Directly matches transfer-time storage (p47).

B - Incorrect: Scheduling reorders requests, rather than providing transfer storage.

C - Incorrect: Caching keeps copies for more efficient access, not specifically transfer mismatch handling.

D - Incorrect: Spooling holds queued device output, a specialized buffer use; no printer/output serialization is specified in the stem.

Primary note proof

Study-note section OS35

Buffering temporarily holds data in transit to absorb speed/transfer-size differences. Caching retains copies to accelerate reuse. Spooling queues jobs for a device such as a printer, usually on disk, so producers need not own it throughout job creation. These have different purposes even if one implementation combines them.

Study-note section OS06

Long-term scheduling admits jobs; medium-term scheduling suspends/swaps them; short-term scheduling selects ready work.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 47 | lines 5-9

> Buffering - Kernel I/O Subsystem maintains a memory area known as buffer that stores data while they are transferred between two devices or between a devices with an application operation. Buffering is done to cope with a speed mismatch between the producer and consumer of a data stream or to adapt between devices that have different data transfer sizes.

Operating System module | PDF page 47 | lines 10-11

> Caching - Kernel maintains cache memory which is region of fast memory that holds copies of data. Access to the cached copy is more efficient than access to the original.

Operating System module | PDF page 47 | lines 12-16

> Spooling and Device Reservation - A spool is a buffer that holds output for a device, such as a printer, that cannot accept interleaved data streams. The spooling system copies the queued spool files to the printer one at a time. In some operating systems, spooling is managed by a system daemon process. In other operating systems, it is handled by an in kernel thread.

Question sources

Mock exam 2 · Question 17

Question 109 | Q01629

Which directory structure describes the commonly presented namespace of modern general-purpose operating systems, with nested directories and absolute or relative paths? Choose the structural category, not every simpler arrangement that might also be implemented.

A. multi-level

B. All choices are correct

C. two level

D. Single level

Hint

Follow namespace structure and distinguish names, underlying files and mounted filesystems.

Answer A: multi-level

A multilevel or tree namespace supports nested directories and paths that can be absolute or relative. That is the structure the notes identify as commonly presented by modern general-purpose systems. A two-level structure separates users but does not express arbitrary nesting, and a single-level structure has only one namespace level. The aggregate therefore includes structures that do not match the requested hierarchical category.

Why each choice is right or wrong

A - Correct: Multilevel describes the hierarchical nested namespace.

B - Incorrect: Single-level and two-level arrangements do not both describe the requested nested tree, so all is incorrect.

C - Incorrect: Two-level separates users without providing the general nested hierarchy asked for.

D - Incorrect: Single-level has no nested directory hierarchy.

Primary note proof

Study-note section OS37

> A single-level directory requires unique names in one namespace. Two-level directories separate users. A tree allows nested directories and absolute/relative paths; links can create shared graphs. Modern general-purpose systems commonly present hierarchical namespaces, but a feature name alone does not prove that every system excludes simpler arrangements.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 6 | lines 19-23

> A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions. Operating System does the following activities for file management.
> Keeps track of information, location, uses, status etc. The collective facilities are often known as file system.

Operating System module | PDF page 54 | lines 5-17

> Portable - Portability means software can works on different types of hardware in same way. Linux kernel and application programs support their installation on any kind of hardware platform.
> Open Source - Linux source code is freely available and it is community-based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
> Multi-User - Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.
> Multiprogramming - Linux is a multiprogramming system means multiple applications can run at same time.
> Hierarchical File System - Linux provides a standard file structure in which system files/ user files are arranged.
> Shell - Linux provides a special interpreter program which can be used to execute

Question sources

Mock exam 2 · Question 18

Question 110 | SM0001

For the first four generations described in NYU's OS-history notes, which pairings incorrectly identify the defining historical characteristic of the named generation? Judge characteristic era, not whether a feature could also exist later. Select all that apply.

A. Second generation -> multiprogramming

B. Fourth generation -> real time systems

C. First generation → manual single-user operation without a separate resident OS

D. Third generation → uniprogrammed batch systems

Hint

Compare the stated historical clues and distinguish origin, characteristic era and technical capability.

Answer A, B, D: Second generation -> multiprogramming; Fourth generation -> real time systems; Third generation → uniprogrammed batch systems

The first four generations in NYU's teaching taxonomy associates the first era with one user and no separate OS, the second with uniprogrammed batch systems, the third with IC-based multiprogramming, and the fourth with personal computers. Thus A, B and D mismatch the characteristic eras, while C fits. OS02 supplies the operating-mode background; the generation associations come from the supplementary university history notes rather than the unchanged master note.

Why each choice is right or wrong

A - Correct: Multiprogramming characterizes the third era, so pairing it with the second is wrong.

B - Correct: The fourth-era heading is personal computers, not the introduction of real-time systems.

C - Incorrect: Manual single-user operation without a separate OS matches the first era.

D - Correct: Uniprogrammed batch characterizes the second era, not the defining third-era advance.

Primary note proof

Study-note section OS-S01

> In the first four generations of the cited NYU teaching taxonomy, the first generation uses vacuum tubes and manual single-user operation without a separate resident OS. The second generation pairs transistors with uniprogrammed batch systems; the resident OS starts successive jobs without being reloaded for every job.

Study-note section OS-S01

> The third generation pairs integrated circuits with multiprogramming. The fourth emphasizes personal computers. These are characteristic-era associations, not claims that each feature existed exclusively in one era.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 7 | lines 20-33

> Time sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time. Time-sharing or multitasking is a logical extension of multiprogramming. Processor's time which is shared among multiple users simultaneously is termed as time-sharing. The main difference between Multi-programmed Batch Systems and Time-Sharing Systems is that in case of multi-programmed batch systems, objective is to maximize processor use, whereas in Time-Sharing Systems objective is to minimize response time.
> Multiple jobs are executed by the CPU by switching between them, but the switches occur so frequently. Thus, the user can receive an immediate response. For example, in a transaction processing, processor execute each user program in a short burst or quantum of computation. That is if n users are present, each user can get time quantum. When the user submits the command, the response time is in few seconds at most.
> Operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time. Computer systems that were designed primarily as batch systems

Operating System module | PDF page 13 | lines 21-25

> 4.3 Multiprogramming
> When two or more programs are residing in memory at the same time, then sharing the processor is referred to the multiprogramming. Multiprogramming assumes a single shared processor. Multiprogramming increases CPU utilization by organizing jobs so that the CPU always has one to execute.

Operating System module | PDF page 7 | lines 5-7

> operating systems are there from the very first computer generation. Operating systems keep evolving over the period of time. Following are few of the important types of operating system which are most commonly used.

Question sources

MOE Computer Science Exit Exam · Question 1-1

National Computer Science Exit Exam · Question 16

National Computer Science Exit Exam · Question 27

National Computer Science Exit Exam

Computer Science Exit Exam practice collection · Question 211

Question 111 | SM0002

Which one of the following is asynchronous in message communication?

A. Non-block receive

B. Blocking receive

C. Blocking send

D. Direct message

Hint

Identify how data moves, which participant waits and what event allows the operation to continue.

Answer A: Nonblocking receive.

OS09 defines nonblocking operations as returning promptly with data, progress or a would-block result. Among the options, nonblocking receive therefore supplies the asynchronous/non-waiting behavior being contrasted. Blocking receive waits for a message, blocking send waits for its defined completion condition, and direct message concerns how endpoints are addressed rather than whether the call blocks.

Why each choice is right or wrong

A - Correct: Nonblocking receive returns promptly instead of waiting until a message arrives, matching this asynchronous contrast.

B - Incorrect: Blocking receive waits for a message rather than providing the non-waiting behavior requested.

C - Incorrect: Blocking send waits for its defined handoff or receipt condition rather than returning promptly.

D - Incorrect: Direct messaging describes endpoint addressing, not whether sending or receiving blocks.

Primary note proof

Study-note section OS09

> Blocking send waits for the required handoff/buffering/receipt condition defined by its API. Blocking receive waits for a message. Nonblocking operations return promptly with data, progress or a would-block result. “Sender blocked until received” identifies blocking send under rendezvous semantics.

Study-note section OS-S04

> Direct message communication names the receiving process explicitly. Indirect communication uses a mailbox or port. This naming distinction is independent of whether a send or receive blocks: naming a destination does not determine synchronization semantics.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 11 | lines 15-19

• Two processes often require data to be transferred between them.
• The both processes can be on the one computer or on different computer but are connected through computer network.
• Communication may be implemented by two methods either by Shared Memory or by Message Passing.

Question sources

MOE Computer Science Exit Exam · Question 1-2

National Computer Science Exit Exam · Question 3

National Computer Science Exit Exam

Computer Science Exit Exam practice collection · Question 212

Question 112 | SM0011

Which one of the following is NOT correct about threads?

A. Threads are the entities scheduled for execution on the CPU

B. It is a group of processes

C. Threads have program counters

D. Threads have registers to hold their working memory

Hint

Compare per-thread execution context with shared process resources and the stated thread mapping.

Answer B: It is a group of processes

PDF p27 defines a thread as a flow within one process, with its own PC and registers; p29 says kernel scheduling occurs on a thread basis. B incorrectly inverts containment by calling a thread a group of processes.

Why each choice is right or wrong

A - Incorrect: Thread-basis scheduling is explicit on p29.

B - Correct: Each thread belongs to exactly one process, not a collection of processes (p27).

C - Incorrect: Own program counter is explicit (p27).

D - Incorrect: Own system registers are explicit (p27); working memory should mean saved execution values rather than all shared process memory.

Primary note proof

Study-note section OS08

A thread has its own PC, registers and stack. Threads in a process share code, address space and usually open-file resources. A process can contain many threads; a thread belongs to one process at a time. Threads are not a group of processes and need not execute sequentially. Independent processes can exist; cooperating ones communicate/share data.

Study-note section OS08

User-level threads are managed by a library; kernel threads are known to the OS. In many-to-one mapping, a blocking kernel call can block all user threads and one kernel execution context cannot use multiple CPUs simultaneously. One-to-one permits another kernel thread to run when one blocks. Many-to-many multiplexes user threads over several kernel threads.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 27 | lines 6-14

A thread is a flow of execution through the process code, with its own program counter, system registers and stack. A thread is also called a light weight process. Threads provide a way to improve application performance through parallelism. Threads represent a software approach to improving performance of operating system by reducing the overhead thread is equivalent to a classical process.
Each thread belongs to exactly one process and no thread can exist outside a process. Each thread represents a separate flow of control. Threads have been successfully used in implementing network servers and web server. They also provide a suitable foundation for parallel execution of applications on shared memory multiprocessors. Following figure shows

Operating System module | PDF page 29 | lines 4-16

In this case, thread management done by the Kernel. There is no thread management code in the application area. Kernel threads are supported directly by the operating system. Any application can be programmed to be multithreaded. All of the threads within an application are supported within a single process.
The Kernel maintains context information for the process as a whole and for individuals' threads within the process. Scheduling by the Kernel is done on a thread basis. The Kernel performs thread creation, scheduling and management in Kernel space. Kernel threads are generally slower to create and manage than the user threads.
Advantages
• Kernel can simultaneously schedule multiple threads from the same process on multiple processes.
• If one thread in a process is blocked, the Kernel can schedule another thread of the same process.

Question sources

MOE Computer Science Exit Exam · Question 3-3

National Computer Science Exit Exam · Question 84

National Computer Science Exit Exam

MOE Computer Science Exit Exam

Computer Science Exit Exam practice collection · Question 213

Question 113 | SM0029

A computer expert would like to know the number of processes that are completed per hour to assess the performance of a machine. Which one of the following best describes the expert's assessment?

A. Waiting time

B. System throughput

C. Turnaround time

D. Response time

Hint

Construct a timeline and distinguish completion rates from individual elapsed-time measures.

Answer B: System throughput.

OS13 defines throughput as completed jobs divided by elapsed time. Counting process completions per hour is exactly that rate. Waiting, turnaround and response are delays associated with an individual process or an average of such delays; they are not completion counts per hour. The stated measurement therefore identifies system throughput without any scheduling trace.

Why each choice is right or wrong

A - Incorrect: Waiting time measures ready-queue delay, not the number of processes completed in an hour.

B - Correct: Completed processes divided by hours elapsed is system throughput here’s definition.

C - Incorrect: Turnaround measures arrival-to-completion duration for a process rather than a system completion rate.

D - Incorrect: Response measures arrival-to-first-start delay, not the number of jobs finished per unit time.

Primary note proof

Study-note section OS13

> For arrival A, first start S and completion C, response=S−A and turnaround=C−A. For a single CPU burst B with no I/O, waiting=C−A−B. With I/O, subtract all CPU and blocked time to isolate ready-queue waiting. Throughput=completed jobs/elapsed time; CPU utilization=busy time/elapsed time. Minimize response/turnaround/waiting under the selected goal and maximize useful throughput/utilization.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 6 | lines 31-35

Control over system performance -- Recording delays between request for a service and response from the system.
Job accounting -- Keeping track of time and resources used by various jobs and users.
Error detecting aids -- Production of dumps, traces, error messages and other debugging and error detecting aids.

Operating System module | PDF page 37 | lines 6-11

Virtual memory is a technique that allows the execution of processes which are not completely available in memory. The main visible advantage of this scheme is that programs can be larger than physical memory. Virtual memory is the separation of user logical memory from physical memory.
This separation allows an extremely large virtual memory to be provided for programmers when only a smaller physical memory is available. Following are the situations, when entire

Operating System module | PDF page 37 | lines 20-26

• Less number of I/O would be needed to load or swap each user program into memory.
• A program would no longer be constrained by the amount of physical memory that is available.
• Each user program could take less physical memory, more programs could be run the same time, with a corresponding increase in CPU utilization and throughput.
Virtual memory is commonly implemented by demand paging. It can also be implemented in a segmentation system. Demand segmentation can also be used to provide virtual memory.

Question sources

MOE Computer Science Exit Exam · Question 9-1

National Computer Science Exit Exam · Question 89

National Computer Science Exit Exam

MOE Computer Science Exit Exam

Computer Science Exit Exam practice collection · Question 214

Question 114 | SM0075

A process is terminated by the operating system due to a reference to non-existing memory. Which one of the following best explains the process termination?

A. Normal exit

B. Error exit

C. Terminated by another process

D. Fatal exit

Hint

Separate creation, image replacement and termination; classify termination by its cause.

Answer D: Fatal exit.

OS07 explicitly identifies a reference to nonexistent/protected memory as a fatal fault. Here the OS terminates the process because of that fault, so this is involuntary fatal termination. It is not the program's voluntary error exit, normal completion or a kill request by another ordinary process. The cause in the stem resolves the otherwise loose exit terminology.

Why each choice is right or wrong

A - Incorrect: Normal exit is voluntary completion, unlike OS-enforced termination for an invalid memory reference.

B - Incorrect: A program-detected error exit is voluntary; the stem specifies an unrecoverable memory-access fault.

C - Incorrect: The stated cause is a fault handled by the OS, not a kill request from another process.

D - Correct: The note explicitly classifies nonexistent-memory access as a fatal fault causing involuntary termination.

Primary note proof

Study-note section OS07

> Normal completion and a program-detected error exit are voluntary; an unrecoverable fault or being killed is involuntary. exit() terminates normally through the runtime; abort() requests abnormal termination. A reference to nonexistent/protected memory is a fatal fault, unlike an application deciding to exit after bad input.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 39 | lines 5-11

> Check an internal table for this process, to determine whether the reference was a valid or it was an invalid memory access.
> Step 2
> If the reference was invalid, terminate the process. If it was valid, but page have not yet brought in, page in the latter.
> Step 3 Find a free frame.
> Step 4 Schedule a disk operation to read the desired page into the newly allocated frame.

Operating System module | PDF page 16 | lines 29-39

> 1. New The process is being created.
> 2.
> Ready The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run.
> 3.
> Running Process instructions are being executed (i.e. The process that is currently being executed).
> 4.

Waiting The process is waiting for some event to occur (such as the completion of an I/O operation).
5. Terminated The process has finished execution.

Question sources

MOE Computer Science Exit Exam · Question 21-3

National Computer Science Exit Exam · Question 1

National Computer Science Exit Exam

MOE Computer Science Exit Exam

Computer Science Exit Exam practice collection · Question 216

Question 115 | SM0207

What is a characteristic feature of round-robin scheduling?

A. It assigns priorities to tasks based on their relative importance

B. It always selects the task with the shortest remaining time

C. It favors short tasks over long tasks

D. It guarantees fairness by providing each task with an equal time slice

Hint

Trace bounded turns in ready-queue order and include the stated overhead and arrivals.

Answer D: It guarantees fairness by providing each task with an equal time slice

PDF p25 gives each process a fixed quantum and preempts after it. D identifies that equal-turn mechanism; it guarantees equal-sized opportunities, not identical lifetime CPU totals for unequal jobs.

Why each choice is right or wrong

A - Incorrect: Priority rank is the p24 policy, not the RR criterion.

B - Incorrect: Remaining-time selection is not RR's fixed quantum.

C - Incorrect: Short-job preference is not RR's fixed quantum rule.

D - Correct: Equal quantum per turn is precisely the p25 definition.

Primary note proof

Study-note section OS18

RR serves a ready queue cyclically for at most quantum q per turn. Unfinished jobs return to the tail; completed/blocked jobs leave.

Study-note section OS17

Priority scheduling chooses the highest-priority ready job.

Study-note section OS16

Non-preemptive SJF/SJN chooses the shortest next burst among currently ready jobs. SRTF reconsiders on arrivals and selects the shortest remaining work; a shorter newcomer can preempt.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 25 | lines 10-14

7.4 Round Robin Scheduling
• Each process is provided a fix time to execute called quantum.
• Once a process is executed for given time period. Process is preempted and other process executes for given time period.
• Context switching is used to save states of preempted processes.

Operating System module | PDF page 24 | lines 10-15

7.3 Priority Based Scheduling
• Each process is assigned a priority. Process with highest priority is to be executed first and so on.
• Processes with same priority are executed on first come first serve basis.
• Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Operating System module | PDF page 23 | lines 23-26

7.2 Shortest Job First (SJF)
• Best approach to minimize waiting time.
• Impossible to implement
• Processer should know in advance how much time process will take.

Question sources

Computer Science Exit Exam · Sene 2016 · Question 2-3

Computer Science Exit Exam practice collection · Question 226

Question 116 | SM0220

Which objective of modern operating systems involves providing a consistent and convenient environment for running applications?

A. Maximizing hardware utilization

B. Ensuring fault tolerance

C. Providing security and protection

D. Providing a user-friendly interface

Hint

Separate broad resource-management goals from individual services or applications.

Answer D: Providing a user-friendly interface

PDF p5 emphasizes a convenient program-execution environment; p14 supplies a user interface with short interactive response. D is the option specifically about user convenience.

Why each choice is right or wrong

A - Incorrect: Utilization concerns efficiency rather than the stated convenient interaction goal.

B - Incorrect: Fault tolerance concerns failure behavior, not ordinary interaction convenience.

C - Incorrect: Security controls authorization (p6), not the named convenience goal.

D - Correct: Interface support is the module's interaction/convenience role (pp5,14).

Primary note proof

Study-note section OS01

> An OS allocates CPU time, memory, storage, devices and communication resources, and provides services to programs. Convenience means making tasks easy through consistent interfaces, files and device abstractions. Efficiency means using resources effectively; it is not synonymous with the fastest response for every process. Application development tools use OS services but application development itself is not a core resource-management function.

Study-note section OS01

> Updates and monitoring help administration, but automatic updates are not a defining property of every OS.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 5 | lines 6-13

> An operating System (OS) is an intermediary between users and computer hardware. It provides users an environment in which a user can execute programs conveniently and efficiently. In technical terms, it is software which manages hardware. An operating System controls the allocation of resources and services such as memory, processors, devices and information.
> 1.1 Definition
> An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.

Operating System module | PDF page 14 | lines 18-23

> Interactivity refers that a User is capable to interact with computer system. Operating system does the following activities related to interactivity.
> • OS provides user an interface to interact with system.
> • OS managers input devices to take inputs from the user. For example, keyboard.

• OS manages output devices to show outputs to the user. For example, Monitor.
• OS Response time needs to be short since the user submits and waits for the result.

Question sources

Computer Science Exit Exam · Sene 2016 · Question 7-1

Computer Science Exit Exam practice collection · Question 224

Question 117 | SM0228

All of the following are scheduling criteria EXCEPT:

A. CPU utilization

B. Min Turnaround Time

C. Max Throughput

D. Max Response Time

Hint

Construct a timeline and distinguish completion rates from individual elapsed-time measures.

Answer D: Maximizing response time is the wrong optimization direction.

OS13 says to minimize response, turnaround and waiting while maximizing useful throughput/utilization under the chosen objective. Thus maximum response time is the mismatched criterion: making first response later worsens responsiveness. The other alternatives name a standard metric or its conventional optimization direction. The question is about preferred direction, not whether response time itself is measurable.

Why each choice is right or wrong

A - Incorrect: CPU utilization is a standard scheduling metric measuring the fraction of elapsed time the CPU is busy.

B - Incorrect: Minimizing turnaround is the conventional goal for reducing arrival-to-completion delay.

C - Incorrect: Maximizing useful throughput is a conventional goal for increasing completed work per unit time.

D - Correct: Response delay should ordinarily be minimized; maximizing it reverses the responsiveness objective.

Primary note proof

Study-note section OS13

For arrival A, first start S and completion C, response=S−A and turnaround=C−A. For a single CPU burst B with no I/O, waiting=C−A−B. With I/O, subtract all CPU and blocked time to isolate ready-queue waiting. Throughput=completed jobs/elapsed time; CPU utilization=busy

time/elapsed time. Minimize response/turnaround/waiting under the selected goal and maximize useful throughput/utilization.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 7 | lines 20-33

Time sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time. Time-sharing or multitasking is a logical extension of multiprogramming. Processor's time which is shared among multiple users simultaneously is termed as time-sharing. The main difference between Multi-programmed Batch Systems and Time-Sharing Systems is that in case of multi-programmed batch systems, objective is to maximize processor use, whereas in Time-Sharing Systems objective is to minimize response time.
Multiple jobs are executed by the CPU by switching between them, but the switches occur so frequently. Thus, the user can receive an immediate response. For example, in a transaction processing, processor execute each user program in a short burst or quantum of computation. That is if n users are present, each user can get time quantum. When the user submits the command, the response time is in few seconds at most.
Operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time. Computer systems that were designed primarily as batch systems

Operating System module | PDF page 13 | lines 21-25

4.3 Multiprogramming
When two or more programs are residing in memory at the same time, then sharing the processor is referred to the multiprogramming. Multiprogramming assumes a single shared processor. Multiprogramming increases CPU utilization by organizing jobs so that the CPU always has one to execute.

Operating System module | PDF page 23 | lines 23-26

7.2 Shortest Job First (SJF)
• Best approach to minimize waiting time.
• Impossible to implement
• Processer should know in advance how much time process will take.

Operating System module | PDF page 37 | lines 20-26

• Less number of I/O would be needed to load or swap each user program into memory.
• A program would no longer be constrained by the amount of physical memory that is available.
• Each user program could take less physical memory, more programs could be run the same time, with a corresponding increase in CPU utilization and throughput.
Virtual memory is commonly implemented by demand paging. It can also be implemented in a segmentation system. Demand segmentation can also be used to provide virtual memory.

Question sources

Computer Science Exit Exam · Sene 2016 · Question 9-3

Computer Science Exit Exam practice collection · Question 225

Question 118 | SM0241

Which umbrella function of modern operating systems allocates and coordinates hardware resources such as CPU time, memory and I/O devices?

A. File Management

B. Resource Management

C. Process Management

D. Security Management

Hint

Separate broad resource-management goals from individual services or applications.

Answer B: Resource Management

An OS allocates CPU time, memory, storage, devices and communication resources. The umbrella term for coordinating these allocations is resource management. File, process and security management cover narrower responsibilities, so none alone names the full scope of the question. Replacing device management with the umbrella category preserves the combined CPU/memory/I/O concept and gives the item one complete answer.

Why each choice is right or wrong

A - Incorrect: File management concerns files and storage abstractions rather than all named resources.

B - Correct: Resource management spans CPU time, memory and I/O devices.

C - Incorrect: Process management covers execution and CPU scheduling but is narrower than the combined scope.

D - Incorrect: Security management controls protection and permissions rather than naming all allocation functions.

Primary note proof

Study-note section OS01

> An OS allocates CPU time, memory, storage, devices and communication resources, and provides services to programs.

Study-note section OS06

> Long-term scheduling admits jobs; medium-term scheduling suspends/swaps them; short-term scheduling selects ready work.

Study-note section OS36

A file has content and metadata such as name, size, owner, permissions and timestamps. open obtains an open-file reference, read/write transfer bytes, seek changes an offset, close releases a reference, truncate changes size and unlink removes a name.

Study-note section OS39

Authentication verifies identity; authorization checks permitted operations.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 6 | lines 4-9

In multiprogramming environment, OS decides which process gets the processor when and how much time. This function is called process scheduling. Operating System does the following activities for processor management. Keeps tracks of processor and status of process. Program responsible for this task is known as traffic controller.
Allocates the processor (CPU) to a process.
De-allocates processor when processor is no longer required.

Operating System module | PDF page 5 | lines 26-36

Memory management refers to management of Primary Memory or Main Memory. Main memory is a large array of words or bytes where each word or byte has its own address. Main memory provides a fast storage that can be access directly by the CPU. So, for a program to be executed, must in the main memory.
Activities Operating System for memory management.
Keeps tracks of primary memory (i.e. what part of it are in use by whom, what part are not in use.)
In multiprogramming, OS decides which process will get memory when and how much.
Allocates the memory when the process requests it to do so.
De-allocates the memory when the process no longer needs it or has been terminated.

Operating System module | PDF page 6 | lines 11-17

OS manages device communication via their respective drivers. Activities of Operating System for device management.
Keeps tracks of all devices. Program responsible for this task is known as the I/O controller.
Decides which process gets the device when and for how much time.
Allocates the device in the efficient way.
De-allocates devices.

Question sources

Computer Science Exit Exam · Sene 2016 · Question 13-3

Computer Science Exit Exam practice collection · Question 223

Question 119 | SM0268

Which scheduling algorithm is commonly associated with the concept of aging, where the priority of a task increases the longer it waits in the ready queue?

A. Round Robin (RR)

B. Shortest Job Next (SJN)

C. Priority Scheduling

D. Shortest Remaining Time First (SRTF)

Hint

Check priority direction, preemption and whether waiting can change effective rank.

Answer C: Priority scheduling.

OS17 describes aging as increasing waiting jobs' effective priority to reduce indefinite postponement. That is a direct adjustment within a priority scheme and matches the stem exactly. RR rotates turns, SJN selects the shortest next burst, and SRTF selects the shortest remaining work. Those policies may be modified, but they are not the named priority-aging mechanism requested.

Why each choice is right or wrong

A - Incorrect: RR cycles ready jobs by quantum rather than defining aging through increasing priority.

B - Incorrect: SJN chooses shortest next bursts; aging refers specifically to adjusting the effective priority of waiting work.

C - Correct: Priority scheduling can raise a waiting task's effective priority through aging to reduce starvation.

D - Incorrect: SRTF compares remaining bursts rather than increasing priority solely because a task has waited longer.

Primary note proof

Study-note section OS17

> Aging increases waiting jobs' effective priority according to a rule, reducing indefinite postponement when the priority scheme and workload permit.

Study-note section OS18

> RR serves a ready queue cyclically for at most quantum q per turn.

Study-note section OS16

> Non-preemptive SJF/SJN chooses the shortest next burst among currently ready jobs. SRTF reconsiders on arrivals and selects the shortest remaining work; a shorter newcomer can preempt.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 24 | lines 10-15

> 7.3 Priority Based Scheduling
> • Each process is assigned a priority. Process with highest priority is to be executed first and so on.
> • Processes with same priority are executed on first come first serve basis.
> • Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Question sources

Computer Science Exit Exam · Sene 2016 · Question 22-2

Computer Science Exit Exam practice collection · Question 222

Question 120 | SM0278

In a system susceptible to deadlock, what does the 'no preemption' condition imply?

A. Resources can be preempted and assigned to other processes

B. Processes cannot request additional resources while holding others

C. Resources cannot be preempted while they are in use

D. Processes must release all resources when blocked

Hint

Trace the resource dependencies and distinguish necessary conditions from evidence of an actual deadlock.

Answer C: Allocated resources cannot be forcibly withdrawn.

OS22 defines no resource preemption as requiring voluntary release by the holder. C states that resources cannot be preempted while in use and therefore matches. A reverses the condition. B and D describe policies attacking hold-and-wait by limiting requests or requiring release; they are different from the no-preemption property itself.

Why each choice is right or wrong

A - Incorrect: Permitting withdrawal and reassignment is resource preemption, directly reversing the condition requested.

B - Incorrect: Forbidding additional requests while holding resources attacks hold-and-wait rather than defining no-preemption.

C - Correct: The holder retains the resource until voluntary release, so it cannot simply be forcibly withdrawn.

D - Incorrect: Requiring release when blocked removes holding-while-waiting; it is not the no-preemption definition.

Primary note proof

Study-note section OS22

> Mutual exclusion allows at most one holder of a nonshareable resource. Hold-and-wait means retaining a resource while requesting another. No resource preemption means release must be voluntary. Circular wait forms a closed chain of such waits.

Study-note section OS24

> “If waiting, release all held resources and try again” removes simultaneous holding while waiting.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

> • Loads a program into memory.
> • Executes the program.
> • Handles program's execution.
> • Provides a mechanism for process synchronization.
> • Provides a mechanism for process communication.
> • Provides a mechanism for deadlock handling.

Question sources

Computer Science Exit Exam · Sene 2016 · Question 25-3

Computer Science Exit Exam practice collection · Question 227

Question 121 | SS106

When a process is in a "Blocked" state waiting for some Input-Output services and afterthe service is completed, it will goes to a state known as:

A. Terminated

B. Suspended

C. Ready

D. Running

Hint

Decide whether the process is eligible, running or waiting, then identify the responsible queue or scheduling role.

Answer C: Ready

After I/O completion the process no longer waits for that event but must obtain CPU allocation, so it becomes Ready (PDF pp16,19-20). Running follows only when the scheduler actually selects it.

Why each choice is right or wrong

A - Incorrect: Completion of an I/O request is not completion of the process.

B - Incorrect: Suspension concerns memory removal/swapping, not an inevitable I/O-completion transition.

C - Correct: Ready means eligible and waiting for processor assignment (p16).

D - Incorrect: It can run after scheduling; I/O completion alone does not allocate CPU (p20).

Primary note proof

Study-note section OS06

> New processes are created/admitted; ready means eligible for CPU but not currently running. Running uses a CPU. Waiting/blocked means unable to proceed until an event; terminated means execution has ended. Running→ready follows time-slice expiry or another preemption. Running→waiting follows blocking I/O. I/O completion normally gives waiting→ready, not a guaranteed immediate running state.

Study-note section OS06

> Long-term scheduling admits jobs; medium-term scheduling suspends/swaps them; short-term scheduling selects ready work.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 16 | lines 29-39

> 1. New The process is being created.
> 2.
> Ready The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run.
> 3.
> Running Process instructions are being executed (i.e. The process that is currently being executed).
> 4.
> Waiting The process is waiting for some event to occur (such as the completion of an I/O operation).
> 5. Terminated The process has finished execution.

Operating System module | PDF page 19 | lines 6-12

> A newly arrived process is put in the ready queue. Processes waits in ready queue for allocating the CPU. Once the CPU is assigned to a process, then that process will execute. While executing the process, any one of the following events can occur.
> • The process could issue an I/O request and then it would be placed in an I/O queue.
> • The process could create new sub process and will wait for its termination.
> • The process could be removed forcibly from the CPU, as a result of interrupt and put back in the ready queue.

Operating System module | PDF page 20 | lines 15-21

It is also called CPU scheduler. Main objective is increasing system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. CPU scheduler selects process among the processes that are ready to execute and allocates CPU to one of them.
Short term scheduler also known as dispatcher, execute most frequently and makes the fine-grained decision of which process to execute next. Short term scheduler is faster than long term scheduler.

Question sources

Computer Science model exam collection · Question 13

Question 122 | SS107

A memory management policy that decisions must be made as to which page or pages are to be supplanted when a memory is full is known as:

A. Cleaning policy

B. Fetch policy

C. Load policy

D. Replacement policy

Hint

Trace every reference, distinguish hits from faults and update the policy state consistently.

Answer D: Replacement policy

Page replacement chooses which memory pages to swap out when allocation requires unavailable free space (PDF p40). D names that victim-selection policy.

Why each choice is right or wrong

A - Incorrect: Cleaning concerns writing dirty contents; it does not identify victim selection in the p40 definition.

B - Incorrect: Fetching concerns bringing data in, rather than selecting a resident victim.

C - Incorrect: Loading concerns admission/bringing in work, rather than selecting pages to evict.

D - Correct: Replacement is explicitly the decision of which pages to swap out (p40).

Primary note proof

Study-note section OS33

Replacement chooses a victim when no suitable free frame is available.

Study-note section OS-S03

A fetch policy decides when contents are brought into memory; demand paging fetches a valid nonresident page when it is accessed. A cleaning policy concerns writing modified contents to

backing storage. A replacement policy chooses which resident page to evict. These decisions can cooperate during one page fault, but they are different decisions.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 40 | lines 7-20

Page replacement algorithms are the techniques using which Operating System decides which memory pages to swap out, write to disk when a page of memory needs to be allocated.
Paging happens whenever a page fault occurs and a free page cannot be used for allocation purpose accounting to reason that pages are not available or the number of free pages is lower than required pages.
When the page that was selected for replacement and was paged out, is referenced again then it has to read in from disk, and this requires for I/O completion. This process determines the quality of the page replacement algorithm: the lesser the time waiting for page-ins, the better is the algorithm. A page replacement algorithm looks at the limited information about accessing the pages provided by hardware, and tries to select which pages should be replaced to minimize the total number of page misses, while balancing it with the costs of primary storage and processor time of the algorithm itself. There are many different page replacement algorithms. We evaluate an algorithm by running it on a particular string of memory reference and computing the number of page faults.

Question sources

Computer Science model exam collection · Question 14

Question 123 | SS122

Which of the following is not a principal event that cause processes to be created?

A. Multiple execution of processes

B. Execution of a process-creation system call by a running process

C. System initialization

D. Auser request to create a new process

Hint

Separate creation, image replacement and termination; classify termination by its cause.

Answer A: Multiple execution is not itself a process-creation event.

OS07 lists initialization, user requests, service requests and a running process's creation call as creation causes. B, C and D match that list. Multiple execution of processes describes work being run, not an explicit request or initialization event creating a new process. Existing processes can execute repeatedly without creating a child, so A is the unsupported creation trigger.

Why each choice is right or wrong

A - Correct: Executing existing processes multiple times is not itself the explicit creation trigger identified here.

B - Incorrect: A running process's creation system call is explicitly one of the listed process-creation causes.

C - Incorrect: System initialization is explicitly listed as a time when processes are created.

D - Incorrect: A user request to create a process is explicitly a valid process-creation cause.

Primary note proof

Study-note section OS07

> Processes arise during initialization, user requests, service requests or a running process's creation call.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 19 | lines 6-12

> A newly arrived process is put in the ready queue. Processes waits in ready queue for allocating the CPU. Once the CPU is assigned to a process, then that process will execute. While executing the process, any one of the following events can occur.
> • The process could issue an I/O request and then it would be placed in an I/O queue.
> • The process could create new sub process and will wait for its termination.
> • The process could be removed forcibly from the CPU, as a result of interrupt and put back in the ready queue.

Operating System module | PDF page 16 | lines 6-9

> A process is a program in execution. The execution of a process must progress in a sequential fashion. Definition of process is following.
> • A process is defined as an entity which represents the basic unit of work to be implemented in the system.

Question sources

Computer Science model exam collection · Question 29

Question 124 | SS125

Which statement about Banker's algorithm is false?

A. It is a CPU scheduling algorithm

B. It uses each process's declared maximum resource claims

C. Its safety test simulates completions to check whether every process can finish

D. It checks a received resource request before granting it

Hint

Compute remaining needs and simulate feasible completions before deciding whether an allocation is safe.

Answer A: It is a CPU scheduling algorithm

Banker's algorithm is a deadlock-avoidance technique, not a CPU scheduling policy. It requires maximum resource claims, computes remaining needs, and simulates possible completions to check for a safe sequence. A request is evaluated after receipt and before being granted, including checks against need and availability. The factual alternatives therefore describe the supported safety procedure, leaving only the CPU-scheduling classification false.

Why each choice is right or wrong

A - Correct: Banker is deadlock avoidance rather than CPU scheduling, so this statement is false.

B - Incorrect: Maximum resource claims are required input, so this statement is true.

C - Incorrect: The safety test repeatedly simulates feasible completions, so this statement is true.

D - Incorrect: A request is checked before granting it, so this statement is true.

Primary note proof

Study-note section OS25

> Avoidance allows requests only if the resulting state remains safe. Banker requires maximum claims. Compute Need=Max−Allocation. Set Work=Available and repeatedly find an unfinished process with Need≤Work componentwise; simulate completion by adding its Allocation to Work. Finishing all gives a safe sequence. Tentatively grant a request only if it is within Need and Available and the safety test succeeds. Unsafe does not mean already deadlocked. Banker is avoidance, not CPU scheduling, detection or prevention by breaking a condition.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

> • Loads a program into memory.
> • Executes the program.
> • Handles program's execution.
> • Provides a mechanism for process synchronization.
> • Provides a mechanism for process communication.
> • Provides a mechanism for deadlock handling.

Question sources

Computer Science model exam collection · Question 32

Question 125 | SS138

A set of processes is blocked: each holds a resource and waits for another resource that only another blocked member of the set can release. No member can proceed without intervention. This state is called:

A. Deadlock

B. Queuing

C. Preemption

D. Overloading

Hint

Trace the resource dependencies and distinguish necessary conditions from evidence of an actual deadlock.

Answer A: Deadlock

The specified set has a closed dependency: every member is waiting for something only another blocked member can provide. None can reach the release operation needed by another, so the state is deadlock. Queuing can occur during ordinary waits that eventually finish; preemption withdraws or reallocates a resource; overload concerns excessive demand. The added dependency condition makes the deadlock interpretation definite.

Why each choice is right or wrong

A - Correct: The closed dependency among blocked members defines deadlock.

B - Incorrect: Ordinary queuing does not require a dependency that prevents every member from progressing.

C - Incorrect: Preemption is resource withdrawal or reassignment rather than this blocked state.

D - Incorrect: Overload does not by itself create the specified closed dependency.

Primary note proof

Study-note section OS22

> A set is deadlocked when each member waits for an event/resource that only another blocked member can provide. Mere waiting for another holder is normal blocking; indefinite unfair delay can be starvation without a cycle of dependencies.

Study-note section OS24

> Prevention constrains how requests are made. Share a resource where safe to remove mutual exclusion; request all needed resources before holding any to remove hold-and-wait; permit safe resource withdrawal/restart to attack no-preemption; impose a global acquisition order to remove circular wait.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 16 | lines 29-39

1. New The process is being created.
2.
Ready The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run.
3.
Running Process instructions are being executed (i.e. The process that is currently being executed).
4.
Waiting The process is waiting for some event to occur (such as the completion of an I/O operation).
5. Terminated The process has finished execution.

Operating System module | PDF page 19 | lines 6-12

A newly arrived process is put in the ready queue. Processes waits in ready queue for allocating the CPU. Once the CPU is assigned to a process, then that process will execute. While executing the process, any one of the following events can occur.
• The process could issue an I/O request and then it would be placed in an I/O queue.
• The process could create new sub process and will wait for its termination.
• The process could be removed forcibly from the CPU, as a result of interrupt and put back in the ready queue.

Operating System module | PDF page 10 | lines 23-28

• Loads a program into memory.
• Executes the program.
• Handles program's execution.
• Provides a mechanism for process synchronization.
• Provides a mechanism for process communication.
• Provides a mechanism for deadlock handling.

Question sources

Computer Science model exam collection · Question 45

Question 126 | SS183

Identify the lowest layer of Android architecture.

A. Application

B. Application Framework

C. Database

D. Linux Kernel

Hint

Use the stated historical or platform scope and distinguish origin from technical capability.

Answer D: Linux kernel.

OS40 directly states that Android's base kernel layer is Linux and places framework and applications above it. Thus Linux Kernel is the lowest offered layer. A database is not identified as the kernel foundation in this model. The answer rests on Android's architecture, not on a separate claim about Android's current market share.

Why each choice is right or wrong

A - Incorrect: Applications sit above the framework and kernel rather than forming Android's base layer.

B - Incorrect: The application framework is explicitly a higher layer than the Linux kernel foundation.

C - Incorrect: A database is not the base kernel layer identified here's Android architecture description.

D - Correct: The note explicitly names Linux as Android's base kernel layer beneath framework and applications.

Primary note proof

Study-note section OS40

> Android's base kernel layer is Linux; its framework and applications are higher layers. AOSP is open source, while a shipped device can include proprietary applications/services.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 54 | lines 24-30

> Hardware layer - Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc.).
> Kernel - Core component of Operating System, interacts directly with hardware, provides low level services to upper layer components.
> Shell - An interface to kernel, hiding complexity of kernel's functions from users. Takes commands from user and executes kernel's functions.
> Utilities - Utility programs giving user most of the functionalities of an operating systems.

Question sources

Computer Science model exam collection · Question 90

Question 127 | AM0654

Which objective specifically concerns efficient allocation of CPU time, memory, storage and devices among competing programs?

A. To provide an interface for users to interact with the hardware

B. To manage and allocate system resources efficiently

C. To ensure data security and protection

D. To optimize program execution speed

Hint

Separate broad resource-management goals from individual services or applications.

Answer B: To manage and allocate system resources efficiently

Resource management allocates CPU time, memory, storage, devices and communications to programs. That directly matches efficient management and allocation in B. User interfaces support convenience, protection controls permitted access, and execution speed is a performance concern. Those are related goals, but the stated question asks specifically for the allocation objective rather than ranking every legitimate OS goal as universally most important.

Why each choice is right or wrong

A - Incorrect: Providing an interface concerns convenience rather than the specific allocation objective.

B - Correct: Efficient management and allocation directly names the requested resource objective.

C - Incorrect: Security and protection concern access control rather than allocation efficiency itself.

D - Incorrect: Program speed is a performance concern and is not equivalent to efficient allocation of all resources.

Primary note proof

Study-note section OS01

> An OS allocates CPU time, memory, storage, devices and communication resources, and provides services to programs. Convenience means making tasks easy through consistent interfaces, files and device abstractions. Efficiency means using resources effectively; it is not synonymous with the fastest response for every process. Application development tools use OS services but application development itself is not a core resource-management function.

Study-note section OS39

> Authentication verifies identity; authorization checks permitted operations.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 5 | lines 6-13

An operating System (OS) is an intermediary between users and computer hardware. It provides users an environment in which a user can execute programs conveniently and efficiently. In technical terms, it is software which manages hardware. An operating System controls the allocation of resources and services such as memory, processors, devices and information.
1.1 Definition
An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.

Question sources

Computer Science MCQ practice collection · Question 654

Question 128 | AM0655

Which of the following is not a function of an operating system?

A. Process management

B. File management

C. Network management

D. Application development

Hint

Separate broad resource-management goals from individual services or applications.

Answer D: Application development is not a core OS resource-management function.

OS01 explicitly distinguishes application-development tools using OS services from application development itself as a core OS function. Processes, files and communication resources are managed by the OS. Therefore D is the exclusion in the intended resource-management scope. Providing compilers, editors or development APIs does not mean the operating system itself performs application development as its defining resource function.

Why each choice is right or wrong

A - Incorrect: Managing execution/process resources belongs to the OS's core resource-management responsibilities.

B - Incorrect: File and storage management are core OS services rather than the requested non-function.

C - Incorrect: Communication/network resource management belongs to the OS responsibilities described in the note.

D - Correct: The note explicitly separates developing applications from the OS functions used by development tools.

Primary note proof

Study-note section OS01

An OS allocates CPU time, memory, storage, devices and communication resources, and provides services to programs. Convenience means making tasks easy through consistent interfaces, files and device abstractions. Efficiency means using resources effectively; it is not synonymous with the fastest response for every process. Application development tools use OS services but application development itself is not a core resource-management function.

Study-note section OS04

Process control, file/device management, information retrieval and communication are service categories.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 6 | lines 4-9

In multiprogramming environment, OS decides which process gets the processor when and how much time. This function is called process scheduling. Operating System does the following activities for processor management. Keeps tracks of processor and status of process. Program responsible for this task is known as traffic controller.
Allocates the processor (CPU) to a process.
De-allocates processor when processor is no longer required.

Operating System module | PDF page 6 | lines 19-23

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions. Operating System does the following activities for file management.
Keeps track of information, location, uses, status etc. The collective facilities are often known as file system.

Operating System module | PDF page 8 | lines 21-26

Network Operating System runs on a server and provides server the capability to manage data, users, groups, security, applications, and other networking functions. The primary purpose of the network operating system is to allow shared file and printer access among multiple computers in a network, typically a local area network (LAN), a private network or to other networks. Examples of network operating systems are Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD.

Operating System module | PDF page 10 | lines 5-15

An Operating System provides services to both the users and to the programs.
• It provides programs, an environment to execute.
• It provides users, services to execute the programs in a convenient manner.
Operating Systems can provide the following common services:
✓ Program execution
✓ I/O operations
✓ File System manipulation
✓ Communication
✓ Error Detection

✓ Resource Allocation
✓ Protection

Question sources

Computer Science MCQ practice collection · Question 655

Question 129 | AM0656

What is the purpose of memory management in an operating system?

A. To provide an interface for users to interact with the computer system

B. To manage the execution of multiple processes simultaneously

C. To allocate and deallocate memory for processes and data

D. To ensure data integrity and security

Hint

Read the cited conceptual rules, state the model, and apply them before comparing the alternatives.

Answer C: To allocate and deallocate memory for processes and data

PDF pp5 and 31 say memory management tracks primary memory, decides allocations, and updates status when memory is freed. C is the specific allocation/deallocation function.

Why each choice is right or wrong

A - Incorrect: User interaction is a separate interface function.

B - Incorrect: Multiple-process execution is process/scheduling work; memory management supports it but does not define execution control.

C - Correct: Allocation and deallocation are the direct memory-management tasks (pp5,31).

D - Incorrect: Protection is supported by base/limit registers, but data integrity/security is too broad to identify the principal allocation role.

Primary note proof

Study-note section OS-S03

Memory management tracks which primary-memory regions are free or in use, assigns memory to processes and data, and deallocates it when no longer needed or when a process terminates. Allocation and deallocation concern memory ownership; CPU selection belongs to scheduling.

Study-note section OS28

In the standard base/limit model, base is that process's physical starting address and limit is its permitted length. Check 0≤logical_offset<limit, then physical=base+offset.

Study-note section OS03

A shell reads/interprets user commands; a GUI is another interface.

Study-note section OS06

Long-term scheduling admits jobs; medium-term scheduling suspends/swaps them; short-term scheduling selects ready work.

Study-note section OS39

Authentication verifies identity; authorization checks permitted operations.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 31 | lines 24-31

Memory management is the functionality of an operating system which handles or manages primary memory. Memory management keeps track of each and every memory location either it is allocated to some process or it is free. It checks how much memory is to be allocated to processes. It decides which process will get memory at what time. It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status.
Memory management provides protection by using two registers, a base register and a limit register. The base register holds the smallest legal physical memory address and the limit register specifies the size of the range. For example, if the base register holds 300000 and the

Operating System module | PDF page 5 | lines 26-36

Memory management refers to management of Primary Memory or Main Memory. Main memory is a large array of words or bytes where each word or byte has its own address. Main memory provides a fast storage that can be access directly by the CPU. So, for a program to be executed, must in the main memory.
Activities Operating System for memory management.
Keeps tracks of primary memory (i.e. what part of it are in use by whom, what part are not in use.)
In multiprogramming, OS decides which process will get memory when and how much.
Allocates the memory when the process requests it to do so.
De-allocates the memory when the process no longer needs it or has been terminated.

Question sources

Computer Science MCQ practice collection · Question 656

Question 130 | AM0657

Which OS component converts generic I/O requests into operations specific to a particular device, rather than naming the kernel as a whole?

A. Kernel

B. Scheduler

C. File system

D. Device drivers

Hint

Separate device hardware, device-specific software and general event or transfer mechanisms.

Answer D: Device drivers

A device driver translates OS I/O requests into the particular operations a device requires. The kernel provides broader resource management and can contain drivers, but the question asks for that specific hardware-facing component. The scheduler selects runnable work and the filesystem organizes file abstractions. Neither performs the general per-device request translation described, making device drivers the precise answer.

Why each choice is right or wrong

A - Incorrect: The kernel is the larger system containing or coordinating the specific component.

B - Incorrect: The scheduler chooses runnable CPU work rather than translating device commands.

C - Incorrect: The filesystem organizes files rather than translating requests for every device type.

D - Correct: Drivers convert OS requests into device-specific operations.

Primary note proof

Study-note section OS34

> A controller is device hardware exposing status, control and data registers. A driver converts OS requests into device-specific operations. Device independence and uniform naming hide brand-specific details; error handling reports/retries supported failures.

Study-note section OS03

> The kernel implements core resource management, protection and system-call handling.

Study-note section OS06

> Long-term scheduling admits jobs; medium-term scheduling suspends/swaps them; short-term scheduling selects ready work.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 47 | lines 19-26

12.5 Device driver
Device driver is a program or routine developed for an I/O device. A device driver implements I/O operations or behaviors on a specific class of devices. For example a system supports one or a number of multiple brands of terminals, all slightly different terminals may have a single terminal driver. In the layered structure of I/O system, device driver lies between interrupt handler and device independent I/O software. The job of a device driver is following.
To accept request from the device independent software above it.
To see to it that the request is executed.

Operating System module | PDF page 45 | lines 28-33

Application I/O Interface represents the structuring techniques and interfaces for the operating system to enable I/O devices to be treated in a standard, uniform way. The actual differences lies kernel level modules called device drivers which are custom tailored to corresponding devices but show one of the standard interfaces to applications. The purpose of the device-driver layer is to hide the differences among device controllers from the I/O subsystem of the kernel, such as the I/O system calls. Following are the characteristics of I/O

Operating System module | PDF page 47 | lines 5-9

Buffering - Kernel I/O Subsystem maintains a memory area known as buffer that stores data while they are transferred between two devices or between a devices with an application operation. Buffering is done to cope with a speed mismatch between the producer and consumer of a data stream or to adapt between devices that have different data transfer sizes.

Question sources

Computer Science MCQ practice collection · Question 657

Question 131 | AM0658

What is the role of the file system in an operating system?

A. To manage system resources and allocate them to processes

B. To provide a user interface for interacting with the computer system

C. To store and organize files on secondary storage devices

D. To schedule and prioritize the execution of processes

Hint

Distinguish a file operation, a resource handle, an offset and file metadata.

Answer C: To store and organize files on secondary storage devices

Files are related information on secondary storage (PDF p48); file management tracks locations and organizes directories (p6). C gives that role.

Why each choice is right or wrong

A - Incorrect: System-wide allocation is broader than organizing files.

B - Incorrect: A user interface is not the file system's stored-information role.

C - Correct: Matches secondary storage and organization (pp6,48).

D - Incorrect: Process execution ordering belongs to schedulers (p20), not file organization.

Primary note proof

Study-note section OS36

> A file has content and metadata such as name, size, owner, permissions and timestamps. open obtains an open-file reference, read/write transfer bytes, seek changes an offset, close releases a reference, truncate changes size and unlink removes a name.

Study-note section OS37

> A single-level directory requires unique names in one namespace. Two-level directories separate users. A tree allows nested directories and absolute/relative paths; links can create shared graphs. Modern general-purpose systems commonly present hierarchical namespaces, but a feature name alone does not prove that every system excludes simpler arrangements.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 48 | lines 5-8

> 13.1 File
> A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks. In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the file's creator and user.

Operating System module | PDF page 6 | lines 19-23

> A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions. Operating System does the following activities for file management.
> Keeps track of information, location, uses, status etc. The collective facilities are often known as file system.

Question sources

Computer Science MCQ practice collection · Question 658

Question 132 | AM0659

Which scheduling algorithm is based on assigning a fixed time slice to each process in a round-robin manner?

A. First-Come, First-Served (FCFS) scheduling

B. Shortest Job Next (SJN) scheduling

C. Priority scheduling

D. Round-Robin scheduling

Hint

Trace bounded turns in ready-queue order and include the stated overhead and arrivals.

Answer D: Round-Robin scheduling

PDF p25 explicitly defines a fixed execution quantum followed by preemption and another process's turn. D is that round-robin policy.

Why each choice is right or wrong

A - Incorrect: FCFS uses arrival order without this specified repeating quantum (p23).

B - Incorrect: SJN selects short jobs using known execution length (p23).

C - Incorrect: Priority uses assigned rank (p24).

D - Correct: Fixed quantum plus repeated turns is the p25 RR definition.

Primary note proof

Study-note section OS18

> RR serves a ready queue cyclically for at most quantum q per turn. Unfinished jobs return to the tail; completed/blocked jobs leave.

Study-note section OS15

> FCFS runs ready bursts in arrival order and ordinarily does not preempt.

Study-note section OS16

> Non-preemptive SJF/SJN chooses the shortest next burst among currently ready jobs.

Study-note section OS17

> Priority scheduling chooses the highest-priority ready job.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 25 | lines 10-14

> 7.4 Round Robin Scheduling
> • Each process is provided a fix time to execute called quantum.
> • Once a process is executed for given time period. Process is preempted and other process executes for given time period.
> • Context switching is used to save states of preempted processes.

Operating System module | PDF page 23 | lines 12-17

> 7.1 First Come First Serve (FCFS)
> • Jobs are executed on first come, first serve basis.
> • Easy to understand and implement.
> • Poor in performance as average wait time is high.
> Wait time of each process is following
> Process Wait Time: Service Time - Arrival Time

Operating System module | PDF page 23 | lines 23-26

7.2 Shortest Job First (SJF)
• Best approach to minimize waiting time.
• Impossible to implement
• Processer should know in advance how much time process will take.

Operating System module | PDF page 24 | lines 10-15

7.3 Priority Based Scheduling
• Each process is assigned a priority. Process with highest priority is to be executed first and so on.
• Processes with same priority are executed on first come first serve basis.
• Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Question sources

Computer Science MCQ practice collection · Question 659

Question 133 | AM0660

What is the primary goal of file-access authorization in an operating system?

A. To ensure the availability of files for authorized users

B. To prevent unauthorized access to files and data

C. To optimize file storage and retrieval performance

D. To minimize the impact of system failures on file operations

Hint

Distinguish identity checks, permission checks and the particular security property at issue.

Answer B: To prevent unauthorized access to files and data

Authorization checks whether a user or other principal is permitted to perform an operation on a file. Preventing unauthorized access is therefore the direct goal. Availability and recovery protect access through failures or disruption, while storage optimization concerns performance. Those are useful objectives, but they are distinct from deciding who may read, write or otherwise access a particular file.

Why each choice is right or wrong

A - Incorrect: Availability keeps files accessible when needed; it is distinct from deciding whether an attempted access is authorized.

B - Correct: Preventing unauthorized access directly follows from checking permitted operations and file rights.

C - Incorrect: Storage and retrieval performance do not determine a caller’s access rights.

D - Incorrect: Limiting failure impact concerns resilience and recovery, rather than the access-permission decision.

Primary note proof

Study-note section OS39

> Authentication verifies identity; authorization checks permitted operations. An ACL lists principals and rights; owner/group/other bits are a simpler policy model, not universal defaults granting all owners full access or everyone read. Read, write, execute, append and delete are different rights, with filesystem-specific rules.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 6 | lines 29-30

> Security -- By means of password and similar other techniques, preventing unauthorized access to programs and data.

Operating System module | PDF page 11 | lines 3-8

> • Program needs to read a file or write a file.
> • The operating system gives the permission to the program for operation on file.
> • Permission varies from read-only, read-write, denied and so on.
> • Operating System provides an interface to the user to create/delete files.
> • Operating System provides an interface to the user to create/delete directories.
> • Operating System provides an interface to create the backup of file system.

Question sources

Computer Science MCQ practice collection · Question 660

Question 134 | AM0661

What is the purpose of virtual memory in an operating system?

A. To provide a virtualized environment for running multiple operating systems

B. To increase the physical memory capacity of a computer system

C. To allow the execution of programs without any memory constraints

D. To create a separate memory space for each running process

Hint

Separate a permitted nonresident access from an illegal access and trace the required handling steps.

Answer D: Provide translated, protected per-process address spaces.

OS32 defines virtual memory as translated/protected address spaces with nonresident portions backed by storage. D is the option matching separate process memory spaces. A describes machine virtualization; B falsely adds physical RAM and C falsely removes all memory constraints. Isolation and translation are the supported purpose, with backing storage providing flexibility within finite resource limits.

Why each choice is right or wrong

A - Incorrect: Running multiple guest OSs describes machine virtualization, explicitly distinguished from process virtual memory.

B - Incorrect: Virtual memory does not physically add RAM; it maps address spaces using existing RAM and backing storage.

C - Incorrect: The note explicitly retains capacity constraints; virtual memory does not make execution memory-unlimited.

D - Correct: Separate translated/protected process address spaces match the virtual-memory purpose described here.

Primary note proof

Study-note section OS32

> Virtual memory provides translated, protected address spaces and allows nonresident portions of a program to be backed by storage. It does not add physical RAM or remove all capacity constraints.

Study-note section OS04

> Virtualization presents a virtual machine environment, often allowing guest operating systems on one host through a hypervisor. It differs from virtual memory, which translates a process's addresses.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 37 | lines 6-11

> Virtual memory is a technique that allows the execution of processes which are not completely available in memory. The main visible advantage of this scheme is that programs can be larger than physical memory. Virtual memory is the separation of user logical memory from physical memory.
> This separation allows an extremely large virtual memory to be provided for programmers when only a smaller physical memory is available. Following are the situations, when entire

Question sources

Computer Science MCQ practice collection · Question 661

Question 135 | AM0662

What is the role of the shell in an operating system?

A. To manage and allocate system resources

B. To provide a graphical user interface for interacting with the system

C. To act as an interface between the user and the operating system kernel

D. To schedule and prioritize the execution of processes

Hint

Distinguish command interfaces, privileged core services and the stages of booting.

Answer C: To act as an interface between the user and the operating system kernel

The shell takes user commands and invokes kernel functions (PDF p54), making C the exact interface relationship. A and D describe other core functions; B imposes a GUI-only restriction absent from the command-interpreter definition.

Why each choice is right or wrong

A - Incorrect: System resources are the kernel/OS management role, not the shell's defining interface.

B - Incorrect: The module describes command interpretation, not a requirement that the shell be graphical.

C - Correct: Directly matches the user-to-kernel interface (p54).

D - Incorrect: CPU ordering is scheduling, not the shell definition.

Primary note proof

Study-note section OS03

> The kernel implements core resource management, protection and system-call handling. A shell reads/interprets user commands; a GUI is another interface. Browsers, office tools and editors are applications.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 54 | lines 18-20

> commands of the operating system. It can be used to do various types of operations,
> call application programs etc.
> Security - Linux provides user security using authentication features like password

Operating System module | PDF page 54 | lines 24-30

> Hardware layer - Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc.).
> Kernel - Core component of Operating System, interacts directly with hardware,
> provides low level services to upper layer components.
> Shell - An interface to kernel, hiding complexity of kernel's functions from users. Takes

commands from user and executes kernel's functions.
Utilities - Utility programs giving user most of the functionalities of an operating systems.

Question sources

Computer Science MCQ practice collection · Question 662

Question 136 | AM0663

What is the primary purpose of process synchronization when processes perform conflicting accesses to shared state?

A. To prevent deadlock situations

B. To ensure fair allocation of system resources

C. To enable parallel execution of multiple processes

D. To coordinate shared-state access and required operation ordering safely

Hint

Identify conflicting shared accesses and the safety or progress property required.

Answer D: To coordinate shared-state access and required operation ordering safely

Conflicting accesses need coordination so interleaving does not corrupt shared state or violate required operation order. Critical-section protection provides mutual exclusion, with progress and bounded waiting required under the stated solution assumptions. Synchronization does not automatically eliminate deadlock; inconsistent lock acquisition can introduce it. It may also serialize work rather than create parallelism, so safe shared-access coordination is the precise primary purpose.

Why each choice is right or wrong

A - Incorrect: Locking can itself deadlock, so universal deadlock prevention is not the primary definition.

B - Incorrect: General resource-allocation fairness is broader than coordinating shared-state operations.

C - Incorrect: Synchronization can serialize conflicting operations and does not itself create parallel execution.

D - Correct: Coordinating shared accesses and required ordering safely captures the primary purpose.

Primary note proof

Study-note section OS10

A race occurs when results depend on uncoordinated interleaving of conflicting accesses. A critical section is code accessing shared state that needs protection, not the resource itself. A

correct exclusion solution requires mutual exclusion, progress and bounded waiting under stated assumptions. Mutual exclusion is a desired safety property; races, starvation and deadlock are possible failures.

Study-note section OS10

Keep critical regions small and ensure every conflicting access uses the same protection. A barrier waits until a defined set of participants reaches a point; a monitor encapsulates protected operations with condition synchronization. Locking can itself introduce deadlock if acquisition order is inconsistent.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

- Loads a program into memory.
- Executes the program.
- Handles program's execution.
- Provides a mechanism for process synchronization.
- Provides a mechanism for process communication.
- Provides a mechanism for deadlock handling.

Question sources

Computer Science MCQ practice collection · Question 663

Question 137 | AM0664

What is the main function of an operating system in terms of convenience?

A. Providing an easy-to-use graphical user interface

B. Optimizing program execution speed

C. Managing system resources efficiently

D. Ensuring data security and protection

Hint

Separate broad resource-management goals from individual services or applications.

Answer A: An easy-to-use interface is the convenience example.

OS01 defines convenience as making tasks easy through consistent interfaces and abstractions. A directly exemplifies that user-facing goal, although a GUI is one possible interface rather than a universal requirement. B and C concern speed/resource efficiency, and D protection. Those are worthwhile OS objectives but do not match the convenience dimension specifically named in the stem.

Why each choice is right or wrong

A - Correct: An easy-to-use GUI is an example of making user tasks convenient through an interface.

B - Incorrect: Optimizing execution speed concerns performance rather than the user-facing convenience dimension requested.

C - Incorrect: Efficient allocation is a resource-use objective, distinguished from convenience in the note.

D - Incorrect: Security/protection is a separate OS objective rather than the interface-usability example of convenience.

Primary note proof

Study-note section OS01

> An OS allocates CPU time, memory, storage, devices and communication resources, and provides services to programs. Convenience means making tasks easy through consistent interfaces, files and device abstractions. Efficiency means using resources effectively; it is not synonymous with the fastest response for every process. Application development tools use OS services but application development itself is not a core resource-management function.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 5 | lines 6-13

> An operating System (OS) is an intermediary between users and computer hardware. It provides users an environment in which a user can execute programs conveniently and efficiently. In technical terms, it is software which manages hardware. An operating System controls the allocation of resources and services such as memory, processors, devices and information.
> 1.1 Definition
> An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.

Operating System module | PDF page 14 | lines 18-23

> Interactivity refers that a User is capable to interact with computer system. Operating system does the following activities related to interactivity.
> • OS provides user an interface to interact with system.
> • OS managers input devices to take inputs from the user. For example, keyboard.
> • OS manages output devices to show outputs to the user. For example, Monitor.
> • OS Response time needs to be short since the user submits and waits for the result.

Question sources

Computer Science MCQ practice collection · Question 664

Question 138 | AM0665

How does an operating system improve efficiency in a computer system?

A. By allocating system resources to processes effectively

B. By providing a wide range of software applications

C. By supporting multiple programming languages

D. By ensuring data integrity and security

Hint

Separate broad resource-management goals from individual services or applications.

Answer A: By allocating system resources to processes effectively

Efficient use follows the module's allocation of processor time and memory to processes (PDF pp5-6,31). A states this mechanism rather than the quantity of applications or languages.

Why each choice is right or wrong

A - Correct: Directly matches resource allocation and scheduling responsibilities.

B - Incorrect: More applications alone does not establish allocation efficiency.

C - Incorrect: Language variety alone does not establish allocation efficiency.

D - Incorrect: Security is a distinct access-control function, not the stated allocation mechanism.

Primary note proof

Study-note section OS01

> An OS allocates CPU time, memory, storage, devices and communication resources, and provides services to programs. Convenience means making tasks easy through consistent interfaces, files and device abstractions. Efficiency means using resources effectively; it is not synonymous with the fastest response for every process.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 5 | lines 6-13

> An operating System (OS) is an intermediary between users and computer hardware. It provides users an environment in which a user can execute programs conveniently and efficiently. In technical terms, it is software which manages hardware. An operating System controls the allocation of resources and services such as memory, processors, devices and information.
> 1.1 Definition
> An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.

Operating System module | PDF page 6 | lines 4-9

In multiprogramming environment, OS decides which process gets the processor when and how much time. This function is called process scheduling. Operating System does the following activities for processor management. Keeps tracks of processor and status of process. Program responsible for this task is known as traffic controller.
Allocates the processor (CPU) to a process.
De-allocates processor when processor is no longer required.

Operating System module | PDF page 31 | lines 24-31

Memory management is the functionality of an operating system which handles or manages primary memory. Memory management keeps track of each and every memory location either it is allocated to some process or it is free. It checks how much memory is to be allocated to processes. It decides which process will get memory at what time. It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status.
Memory management provides protection by using two registers, a base register and a limit register. The base register holds the smallest legal physical memory address and the limit register specifies the size of the range. For example, if the base register holds 300000 and the

Question sources

Computer Science MCQ practice collection · Question 665

Question 139 | AM0666

What does it mean for an operating system to have the ability to evolve?

A. It can adapt to changing hardware configurations

B. It can support multiple operating systems simultaneously

C. It can run legacy software without compatibility issues

D. It can automatically update itself with new features and bug fixes

Hint

Separate broad resource-management goals from individual services or applications.

Answer A: Adapt to changing hardware configurations.

OS01 defines ability to evolve as adding hardware support, features and repairs through maintainable interfaces/modules. Adaptation to hardware configurations is therefore the direct example. Running multiple guest OSs describes virtualization. Backward compatibility preserves supported behavior rather than guaranteeing every legacy binary, and automatic updating is not a defining property of all operating systems.

Why each choice is right or wrong

A - Correct: Adding support for changing hardware is a direct example of maintainable OS evolution.

B - Incorrect: Supporting multiple guest operating systems describes virtualization rather than the definition of ability to evolve.

C - Incorrect: The absolute absence of compatibility issues is not guaranteed by evolution or supported backward compatibility.

D - Incorrect: Automatic updates may be provided, but the note explicitly rejects them as a defining property of every OS.

Primary note proof

Study-note section OS01

> Ability to evolve means adding hardware support, features and repairs through maintainable interfaces/modules. Backward compatibility preserves supported older application behavior; it is not a promise that every old binary works.

Study-note section OS04

> Virtualization presents a virtual machine environment, often allowing guest operating systems on one host through a hypervisor.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 8 | lines 28-31

> • Centralized servers are highly stable.
> • Security is server managed.
> • Upgrades to new technologies and hardware can be easily integrated into the system.
> • Remote access to servers is possible from different locations and types of systems.

Operating System module | PDF page 54 | lines 5-17

> Portable - Portability means software can works on different types of hardware in same way. Linux kernel and application programs support their installation on any kind of hardware platform.
> Open Source - Linux source code is freely available and it is community-based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
> Multi-User - Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.
> Multiprogramming - Linux is a multiprogramming system means multiple applications can run at same time.
> Hierarchical File System - Linux provides a standard file structure in which system files/ user files are arranged.
> Shell - Linux provides a special interpreter program which can be used to execute

Question sources

Computer Science MCQ practice collection · Question 666

Question 140 | AM0667

Which file-management feature directly helps users locate files and navigate among organized groups of files?

A. By providing advanced encryption techniques for file security

B. By allowing users to organize files in a hierarchical structure

C. By automatically backing up files to remote servers

D. By optimizing file storage to maximize disk space utilization

Hint

Follow namespace structure and distinguish names, underlying files and mounted filesystems.

Answer B: By allowing users to organize files in a hierarchical structure

A hierarchical directory tree gives users nested groups and paths for locating files and navigating between directories. This directly supplies the requested organizational convenience. Encryption protects confidentiality, backups preserve additional copies for recovery, and storage optimization concerns space utilization. Those functions may also be useful, but they do not themselves define the navigable directory organization asked for here.

Why each choice is right or wrong

A - Incorrect: Encryption protects confidentiality; it does not supply nested directory paths for navigation.

B - Correct: Hierarchical directories organize files into nested groups reachable through paths.

C - Incorrect: Remote backup concerns additional stored copies, rather than the structure used to navigate the working file namespace.

D - Incorrect: Maximizing disk-space utilization is an allocation goal rather than a directory-navigation feature.

Primary note proof

Study-note section OS37

> A single-level directory requires unique names in one namespace. Two-level directories separate users. A tree allows nested directories and absolute/relative paths; links can create shared graphs.

Study-note section OS39

> Encryption protects confidentiality under correct key use but does not automatically provide integrity.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 6 | lines 19-23

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions. Operating System does the following activities for file management.
Keeps track of information, location, uses, status etc. The collective facilities are often known as file system.

Operating System module | PDF page 54 | lines 5-17

Portable - Portability means software can works on different types of hardware in same way. Linux kernel and application programs support their installation on any kind of hardware platform.
Open Source - Linux source code is freely available and it is community-based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
Multi-User - Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.
Multiprogramming - Linux is a multiprogramming system means multiple applications can run at same time.
Hierarchical File System - Linux provides a standard file structure in which system files/ user files are arranged.
Shell - Linux provides a special interpreter program which can be used to execute

Question sources

Computer Science MCQ practice collection · Question 667

Question 141 | AM0668

What is the role of device drivers in an operating system's convenience factor?

A. They allow users to customize the appearance of the graphical user interface.

B. They provide compatibility with a wide range of hardware devices.

C. They optimize system performance for resource-intensive applications.

D. They enhance data security and encryption for sensitive files.

Hint

Separate device hardware, device-specific software and general event or transfer mechanisms.

Answer B: They provide compatibility with a wide range of hardware devices.

Drivers hide peculiarities of device hardware (PDF p10); p47 even explains a terminal driver supporting differing brands. Standard interfaces on p45 let applications use differing controllers, matching B.

Why each choice is right or wrong

A - Incorrect: GUI appearance is not the device-specific I/O abstraction.

B - Correct: Device compatibility through hardware-specific implementation and common interfaces matches pp10,45,47.

C - Incorrect: Application performance may benefit, but optimizing all resource-intensive applications is not the driver definition.

D - Incorrect: Encryption is not the driver's general stated I/O responsibility.

Primary note proof

Study-note section OS34

A controller is device hardware exposing status, control and data registers. A driver converts OS requests into device-specific operations. Device independence and uniform naming hide brand-specific details; error handling reports/retries supported failures.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 30-37

I/O subsystem comprised of I/O devices and their corresponding driver software. Drivers hides the peculiarities of specific hardware devices from the user as the device driver knows the peculiarities of the specific device.
Operating System manages the communication between user and device drivers.
The major activities of an operating system with respect to I/O Operation.
• I/O operation means read or write operation with any file or any specific I/O device.
• Program may require any I/O device while running.
• Operating system provides the access to the required I/O device when required.

Operating System module | PDF page 47 | lines 19-26

12.5 Device driver
Device driver is a program or routine developed for an I/O device. A device driver implements I/O operations or behaviors on a specific class of devices. For example a system supports one or a number of multiple brands of terminals, all slightly different terminals may have a single terminal driver. In the layered structure of I/O system, device driver lies between interrupt handler and device independent I/O software. The job of a device driver is following.
To accept request from the device independent software above it.
To see to it that the request is executed.

Operating System module | PDF page 45 | lines 28-33

Application I/O Interface represents the structuring techniques and interfaces for the operating system to enable I/O devices to be treated in a standard, uniform way. The actual differences lies kernel level modules called device drivers which are custom tailored to corresponding devices but show one of the standard interfaces to applications. The purpose of the device-driver layer is to hide the differences among device controllers from the I/O subsystem of the kernel, such as the I/O system calls. Following are the characteristics of I/O

Question sources

Computer Science MCQ practice collection · Question 668

Question 142 | AM0669

How does an operating system improve efficiency in terms of process management?

A. By providing a comprehensive programming environment

B. By executing processes in parallel on multiple CPUs

C. By automatically compressing files to reduce storage space

D. By enabling users to schedule program execution at specific times

Hint

Compare interaction, CPU sharing and deadline requirements under the stated system model.

Answer B: By executing processes in parallel on multiple CPUs

PDF p29 says the kernel can simultaneously schedule threads from one process across multiple processors, while p27 explains parallel performance benefits. B is the relevant offered mechanism when suitable parallel work and CPUs exist.

Why each choice is right or wrong

A - Incorrect: A programming environment does not itself describe parallel process execution.

B - Correct: Scheduling work across multiple CPUs exploits parallelism (pp27,29).

C - Incorrect: File compression is unrelated to CPU process management here.

D - Incorrect: User-chosen execution times are not the multiprocessor-efficiency mechanism.

Primary note proof

Study-note section OS02

> A single CPU interleaves execution; multiple CPUs can execute work truly in parallel. Shared-memory multiprocessors do not require each CPU to have private main memory.

Study-note section OS08

> In many-to-one mapping, a blocking kernel call can block all user threads and one kernel execution context cannot use multiple CPUs simultaneously. One-to-one permits another kernel thread to run when one blocks. Many-to-many multiplexes user threads over several kernel threads.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 29 | lines 4-16

> In this case, thread management done by the Kernel. There is no thread management code in the application area. Kernel threads are supported directly by the operating system. Any application can be programmed to be multithreaded. All of the threads within an application are supported within a single process.
> The Kernel maintains context information for the process as a whole and for individuals' threads within the process. Scheduling by the Kernel is done on a thread basis. The Kernel

performs thread creation, scheduling and management in Kernel space. Kernel threads are generally slower to create and manage than the user threads.
Advantages
• Kernel can simultaneously schedule multiple threads from the same process on multiple processes.
• If one thread in a process is blocked, the Kernel can schedule another thread of the same process.

Operating System module | PDF page 27 | lines 6-14

A thread is a flow of execution through the process code, with its own program counter, system registers and stack. A thread is also called a light weight process. Threads provide a way to improve application performance through parallelism. Threads represent a software approach to improving performance of operating system by reducing the overhead thread is equivalent to a classical process.
Each thread belongs to exactly one process and no thread can exist outside a process. Each thread represents a separate flow of control. Threads have been successfully used in implementing network servers and web server. They also provide a suitable foundation for parallel execution of applications on shared memory multiprocessors. Following figure shows

Question sources

Computer Science MCQ practice collection · Question 669

Question 143 | AM0670

In the context of operating systems, what is the meaning of scalability?

A. The ability to run on different hardware platforms

B. The ability to support multiple programming languages

C. The ability to handle increasing workloads and resources

D. The ability to provide a user-friendly interface

Hint

Separate broad resource-management goals from individual services or applications.

Answer C: Handle increasing workloads and resources.

OS01 directly defines scalability as handling greater workloads/resources. Running on different hardware platforms is portability, and providing an easy interface is convenience. Programming-language support does not itself establish either increasing capacity or workload handling. The wording in C matches the supplied definition without assuming scalability guarantees limitless growth or unchanged performance.

Why each choice is right or wrong

A - Incorrect: Adapting to other hardware platforms is portability, explicitly distinguished from scalability.

B - Incorrect: Supporting programming languages does not define the ability to handle growing workload or resources.

C - Correct: Handling increasing workloads and resources directly matches the note's definition of scalability.

D - Incorrect: A user-friendly interface concerns convenience rather than scaling capacity as demand grows.

Primary note proof

Study-note section OS01

> Scalability means handling greater workload/resources; portability means adapting to other platforms.

Study-note section OS01

> Convenience means making tasks easy through consistent interfaces, files and device abstractions.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 54 | lines 5-17

> Portable - Portability means software can works on different types of hardware in same way. Linux kernel and application programs support their installation on any kind of hardware platform.
> Open Source - Linux source code is freely available and it is community-based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
> Multi-User - Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.
> Multiprogramming - Linux is a multiprogramming system means multiple applications can run at same time.
> Hierarchical File System - Linux provides a standard file structure in which system files/ user files are arranged.
> Shell - Linux provides a special interpreter program which can be used to execute

Question sources

Computer Science MCQ practice collection · Question 670

Question 144 | AM0671

How does an operating system ensure convenience in terms of software installation and management?

A. By providing an intuitive software development environment

B. By offering automatic software updates and patches

C. By optimizing the execution speed of software applications

D. By allowing users to customize the appearance of the graphical user interface

Hint

Separate broad resource-management goals from individual services or applications.

Answer B: Automatic updates and patches can make software maintenance convenient.

OS01 defines convenience as making tasks easy and notes that updates help administration. Among the alternatives, offering automatic updates and patches directly reduces routine software-management effort. This is an example of a convenience feature, not a universal guarantee that every OS updates automatically. Development environments, speed optimization and appearance customization address different activities from installation/maintenance in the stem.

Why each choice is right or wrong

A - Incorrect: A development environment assists creating software rather than directly providing installation and maintenance convenience.

B - Correct: Automating updates and patches can reduce routine software-management work, though the feature is not universal.

C - Incorrect: Execution-speed optimization concerns performance, not the installation/maintenance convenience specifically requested.

D - Incorrect: Customizing GUI appearance changes presentation rather than automating software installation or maintenance tasks.

Primary note proof

Study-note section OS01

> Convenience means making tasks easy through consistent interfaces, files and device abstractions.

Study-note section OS01

> Updates and monitoring help administration, but automatic updates are not a defining property of every OS.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 8 | lines 28-31

• Centralized servers are highly stable.
• Security is server managed.
• Upgrades to new technologies and hardware can be easily integrated into the system.
• Remote access to servers is possible from different locations and types of systems.

Operating System module | PDF page 54 | lines 5-17

Portable - Portability means software can works on different types of hardware in same way. Linux kernel and application programs support their installation on any kind of hardware platform.
Open Source - Linux source code is freely available and it is community-based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
Multi-User - Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.
Multiprogramming - Linux is a multiprogramming system means multiple applications can run at same time.
Hierarchical File System - Linux provides a standard file structure in which system files/ user files are arranged.
Shell - Linux provides a special interpreter program which can be used to execute

Question sources

Computer Science MCQ practice collection · Question 671

Question 145 | AM0672

What does it mean for an operating system to provide fault tolerance?

A. It continues correct service despite the specified failures it is designed to tolerate

B. It can handle multiple user requests simultaneously.

C. It can optimize system performance for resource-intensive tasks.

D. It can provide a secure environment for sensitive data.

Hint

Separate broad resource-management goals from individual services or applications.

Answer A: It continues correct service despite the specified failures it is designed to tolerate

Fault tolerance means that correct service continues despite specified failures within the system's fault model. Recovery restores service after a failure and may be useful without providing uninterrupted correct service. Multiple-user handling concerns concurrency, resource-intensive performance concerns efficiency, and sensitive-data protection concerns security. The revised continuation option therefore states the specific property rather than substituting quick recovery for fault tolerance.

Why each choice is right or wrong

A - Correct: Continued correct service despite specified faults is the fault-tolerance property.

B - Incorrect: Simultaneous user requests concern concurrency rather than surviving failures.

C - Incorrect: Performance optimization does not define continued service under faults.

D - Incorrect: Security protects data but does not itself define fault tolerance.

Primary note proof

Study-note section OS01

> Fault tolerance continues correct service despite specified failures; recovery restores service afterward.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 8 | lines 15-16

> • If one site fails in a distributed system, the remaining sites can potentially continue operating.

Operating System module | PDF page 47 | lines 17-18

> Error Handling - An operating system that uses protected memory can guard against many kinds of hardware and application errors.

Question sources

Computer Science MCQ practice collection · Question 672

Question 146 | AM0673

How does an operating system support the ability to evolve in terms of software compatibility?

A. By providing a wide range of pre-installed software applications

B. By offering backward compatibility for legacy software

C. By optimizing the execution speed of software applications

D. By allowing users to customize system settings and preferences

Hint

Separate broad resource-management goals from individual services or applications.

Answer B: Backward compatibility for supported legacy software.

OS01 defines backward compatibility as preserving supported older application behavior. That is the software-compatibility aspect of evolution asked here. Preinstalled applications, execution speed and preference customization do not establish that older software's supported behavior remains available. The answer must retain the support qualification; it does not promise all legacy binaries run without any compatibility issues.

Why each choice is right or wrong

A - Incorrect: Preinstalling many applications does not establish compatibility with older applications or their supported behavior.

B - Correct: Backward compatibility preserves supported legacy behavior, directly matching the software-compatibility aspect of evolution.

C - Incorrect: Faster execution does not by itself preserve interfaces or behavior required by legacy software.

D - Incorrect: Customizing preferences changes configuration rather than establishing compatibility with older application behavior.

Primary note proof

Study-note section OS01

> Backward compatibility preserves supported older application behavior; it is not a promise that every old binary works.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 53 | lines 5-7

> Linux is one of popular version of UNIX operating System. It is open source as its source code is freely available. It is free to use. Linux was designed considering UNIX compatibility. Its functionality list is quite similar to that of UNIX.

Operating System module | PDF page 54 | lines 5-17

> Portable - Portability means software can works on different types of hardware in same way. Linux kernel and application programs support their installation on any kind of hardware platform.
> Open Source - Linux source code is freely available and it is community-based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
> Multi-User - Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.
> Multiprogramming - Linux is a multiprogramming system means multiple applications can run at same time.
> Hierarchical File System - Linux provides a standard file structure in which system files/ user files are arranged.
> Shell - Linux provides a special interpreter program which can be used to execute

Question sources

Computer Science MCQ practice collection · Question 673

Question 147 | AM0674

What is deadlock in the standard process/resource model?

A. A state in which each member of a blocked set waits for an event or resource that only another blocked member can provide

B. It is a state where a process is terminated abruptly.

C. It is a state where multiple processes are executing concurrently.

D. It is a state where the operating system crashes due to a software bug.

Hint

Trace the resource dependencies and distinguish necessary conditions from evidence of an actual deadlock.

Answer A: A state in which each member of a blocked set waits for an event or resource that only another blocked member can provide

Deadlock describes a blocked set whose members depend on one another for the events or resources required to continue. Because each provider is itself blocked within the same dependency, normal execution cannot resolve the waits. A process being terminated, multiple processes executing, or an OS crash describes a different condition. The revised definition also distinguishes deadlock from starvation caused by unfair repeated bypass.

Why each choice is right or wrong

A - Correct: This states the closed dependency that defines the deadlocked set.

B - Incorrect: Abrupt termination ends execution rather than defining blocked dependencies.

C - Incorrect: Concurrent execution is normal and does not imply deadlock.

D - Incorrect: An OS crash is a different failure; deadlocked processes can exist while the OS remains active.

Primary note proof

Study-note section OS22

> A set is deadlocked when each member waits for an event/resource that only another blocked member can provide. Mere waiting for another holder is normal blocking; indefinite unfair delay can be starvation without a cycle of dependencies.

Study-note section OS06

> New processes are created/admitted; ready means eligible for CPU but not currently running. Running uses a CPU. Waiting/blocked means unable to proceed until an event; terminated means execution has ended.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

• Loads a program into memory.
• Executes the program.
• Handles program's execution.
• Provides a mechanism for process synchronization.
• Provides a mechanism for process communication.
• Provides a mechanism for deadlock handling.

Operating System module | PDF page 16 | lines 29-39

1. New The process is being created.
2.
Ready The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run.
3.
Running Process instructions are being executed (i.e. The process that is currently being executed).
4.
Waiting The process is waiting for some event to occur (such as the completion of an I/O operation).
5. Terminated The process has finished execution.

Question sources

Computer Science MCQ practice collection · Question 674

Question 148 | AM0675

Which of the following conditions must be present for deadlock to occur?

A. Mutual exclusion, hold and wait, no preemption, and circular wait.

B. Mutual exclusion, hold and wait, preemption, and circular wait.

C. Mutual inclusion, hold and wait, no preemption, and circular wait.

D. Mutual inclusion, hold and wait, preemption, and circular wait.

Hint

Trace the resource dependencies and distinguish necessary conditions from evidence of an actual deadlock.

Answer A: Mutual exclusion, hold-and-wait, no-preemption and circular wait.

OS22 lists the four necessary conditions in the standard reusable-resource model. A preserves the exact set: exclusion, holding while requesting more, no forced resource withdrawal, and a wait cycle. B introduces preemption in place of no-preemption; C and D replace exclusion with inclusion. These are not harmless wording changes because they remove or reverse the relevant restrictions.

Why each choice is right or wrong

A - Correct: This option states all four standard necessary conditions with their correct meanings and polarity.

B - Incorrect: It substitutes preemption for no-preemption, reversing the required resource-withdrawal restriction.

C - Incorrect: It substitutes mutual inclusion for mutual exclusion, losing the nonshareable-resource condition.

D - Incorrect: It reverses both exclusion and no-preemption, so it is not the required condition set.

Primary note proof

Study-note section OS22

> Mutual exclusion allows at most one holder of a nonshareable resource. Hold-and-wait means retaining a resource while requesting another. No resource preemption means release must be voluntary. Circular wait forms a closed chain of such waits.

Study-note section OS22

> All four are necessary in the standard reusable-resource model. Their availability as general system policies does not prove a deadlock exists at this instant; inspect actual allocations/requests.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

> • Loads a program into memory.
> • Executes the program.
> • Handles program's execution.
> • Provides a mechanism for process synchronization.
> • Provides a mechanism for process communication.
> • Provides a mechanism for deadlock handling.

Question sources

Computer Science MCQ practice collection · Question 675

Question 149 | AM0676

What is mutual exclusion in the context of deadlock?

A. It is the condition where multiple processes can access the same resource simultaneously.

B. It is the condition where a resource can be accessed by only one process at a time.

C. It is the condition where a process holds a resource while waiting for another resource.

D. It is the condition where a process waits indefinitely for a resource.

Hint

Trace the resource dependencies and distinguish necessary conditions from evidence of an actual deadlock.

Answer B: At most one process holds the nonshareable resource at a time.

OS22 defines mutual exclusion as allowing at most one holder of a nonshareable resource. B states that one-at-a-time condition. A describes concurrent sharing, C describes holding while requesting another resource, and D describes indefinite waiting without establishing exclusivity. The definition concerns access safety, not a guarantee that every waiting process eventually obtains the resource.

Why each choice is right or wrong

A - Incorrect: Simultaneous access by several processes contradicts the exclusive-holder condition for a nonshareable resource.

B - Correct: Only one process at a time may hold or use the nonshareable resource.

C - Incorrect: Retaining one resource while requesting another defines hold-and-wait rather than mutual exclusion.

D - Incorrect: Indefinite waiting describes a possible failure outcome and does not define exclusive resource possession.

Primary note proof

Study-note section OS22

> Mere waiting for another holder is normal blocking; indefinite unfair delay can be starvation without a cycle of dependencies. Mutual exclusion allows at most one holder of a nonshareable resource. Hold-and-wait means retaining a resource while requesting another. No resource preemption means release must be voluntary. Circular wait forms a closed chain of such waits.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

> • Loads a program into memory.
> • Executes the program.
> • Handles program's execution.
> • Provides a mechanism for process synchronization.
> • Provides a mechanism for process communication.
> • Provides a mechanism for deadlock handling.

Question sources

Computer Science MCQ practice collection · Question 676

Question 150 | AM0677

What is hold and wait in the context of deadlock?

A. It is the condition where a process holds a resource while waiting for another resource.

B. It is the condition where a process waits indefinitely for a resource.

C. It is the condition where multiple processes can access the same resource simultaneously.

D. It is the condition where a resource can be accessed by only one process at a time.

Hint

Trace the resource dependencies and distinguish necessary conditions from evidence of an actual deadlock.

Answer A: Retain a resource while waiting for another.

OS22 directly defines hold-and-wait as retaining a resource while requesting another. A contains both parts of that definition. Merely waiting indefinitely does not establish a retained allocation, simultaneous sharing contradicts exclusion, and one-at-a-time access describes mutual exclusion rather than a process holding one resource while requesting more.

Why each choice is right or wrong

A - Correct: The process simultaneously retains an allocated resource and waits to acquire a further one.

B - Incorrect: Indefinite waiting does not establish that the process still holds another resource.

C - Incorrect: Simultaneous shared access is not the retained-allocation-plus-request condition defining hold-and-wait.

D - Incorrect: One process accessing a resource at a time defines mutual exclusion, not hold-and-wait.

Primary note proof

Study-note section OS22

> Mere waiting for another holder is normal blocking; indefinite unfair delay can be starvation without a cycle of dependencies. Mutual exclusion allows at most one holder of a nonshareable resource. Hold-and-wait means retaining a resource while requesting another. No resource preemption means release must be voluntary. Circular wait forms a closed chain of such waits.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

> • Loads a program into memory.
> • Executes the program.
> • Handles program's execution.
> • Provides a mechanism for process synchronization.

- Provides a mechanism for process communication.
- Provides a mechanism for deadlock handling.

Question sources

Computer Science MCQ practice collection · Question 677

Question 151 | AM0678

What is preemption in the context of deadlock prevention?

A. It is the condition where a process holds a resource while waiting for another resource.

B. It is the condition where a process waits indefinitely for a resource.

C. It is the condition where multiple processes can access the same resource simultaneously.

D. It is the condition where a resource can be forcibly taken from a process.

Hint

Identify the resource-request restriction and which necessary condition it removes.

Answer D: A resource may be forcibly taken from its holder.

OS22 defines no-preemption as voluntary release only, while OS24 permits safe withdrawal/restart as a way to attack that restriction. Preemption is the opposite permission: a resource can be forcibly reclaimed. D directly states it. The answer concerns resources that can safely be withdrawn, not a claim that every device or lock can be preempted without corruption.

Why each choice is right or wrong

A - Incorrect: Holding one resource while waiting for another describes hold-and-wait rather than forced resource withdrawal.

B - Incorrect: Indefinite waiting is a possible outcome, not the permission to reclaim a held resource.

C - Incorrect: Concurrent sharing concerns resource shareability rather than withdrawing an existing allocation from its holder.

D - Correct: Forcibly taking a held resource is resource preemption, subject to safe withdrawal/restart support.

Primary note proof

Study-note section OS24

Prevention constrains how requests are made. Share a resource where safe to remove mutual exclusion; request all needed resources before holding any to remove hold-and-wait; permit safe resource withdrawal/restart to attack no-preemption; impose a global acquisition order to remove circular wait. Not every physical resource is shareable or safely preemptable.

Study-note section OS22

Mere waiting for another holder is normal blocking; indefinite unfair delay can be starvation without a cycle of dependencies. Mutual exclusion allows at most one holder of a nonshareable resource. Hold-and-wait means retaining a resource while requesting another. No resource preemption means release must be voluntary. Circular wait forms a closed chain of such waits.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

• Loads a program into memory.
• Executes the program.
• Handles program's execution.
• Provides a mechanism for process synchronization.
• Provides a mechanism for process communication.
• Provides a mechanism for deadlock handling.

Operating System module | PDF page 19 | lines 6-12

A newly arrived process is put in the ready queue. Processes waits in ready queue for allocating the CPU. Once the CPU is assigned to a process, then that process will execute. While executing the process, any one of the following events can occur.
• The process could issue an I/O request and then it would be placed in an I/O queue.
• The process could create new sub process and will wait for its termination.
• The process could be removed forcibly from the CPU, as a result of interrupt and put back in the ready queue.

Question sources

Computer Science MCQ practice collection · Question 678

Question 152 | AM0679

What is circular wait in the context of deadlock?

A. It is the condition where a process waits indefinitely for a resource.

B. It is the condition where multiple processes can access the same resource simultaneously.

C. It is the condition where a process holds a resource while waiting for another resource.

D. It is the condition where a circular chain of processes exists, each holding a resource requested
by the next process in the chain.

Hint

Trace the resource dependencies and distinguish necessary conditions from evidence of an actual deadlock.

Answer D: A closed chain of resource waits.

OS22 defines circular wait as a closed chain of waits. D specifies that each process holds a resource requested by the next in a circular chain, which returns the dependency to the starting participant. C describes only one hold-and-wait relationship, which need not be cyclic. Indefinite waiting and simultaneous sharing likewise fail to establish a closed chain.

Why each choice is right or wrong

A - Incorrect: Indefinite waiting does not identify a closed cycle of dependencies among resource holders.

B - Incorrect: Simultaneous resource sharing is not a chain of blocked requests and does not define circular wait.

C - Incorrect: One process holding while requesting another establishes hold-and-wait but not necessarily a cycle.

D - Correct: The circular chain of holdings and requests explicitly forms the required closed wait dependency.

Primary note proof

Study-note section OS22

> Mere waiting for another holder is normal blocking; indefinite unfair delay can be starvation without a cycle of dependencies. Mutual exclusion allows at most one holder of a nonshareable resource. Hold-and-wait means retaining a resource while requesting another. No resource preemption means release must be voluntary. Circular wait forms a closed chain of such waits.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

> • Loads a program into memory.
> • Executes the program.
> • Handles program's execution.
> • Provides a mechanism for process synchronization.
> • Provides a mechanism for process communication.
> • Provides a mechanism for deadlock handling.

Question sources

Computer Science MCQ practice collection · Question 679

Question 153 | AM0680

Which listed policy does not actively prevent an unsafe allocation or detect and recover from an existing deadlock? Interpret “deadlock acceptance” as deliberately ignoring sufficiently rare deadlocks.

A. Deadlock avoidance

B. Deadlock detection and recovery

C. Deadlock prevention

D. Deadlock acceptance

Hint

Separate handling before allocation from actions taken after detection, and consider the cost of checking.

Answer D: Deadlock acceptance

Prevention constrains requests to break a necessary condition; avoidance keeps tentative allocations safe; detection and recovery identifies and resolves an existing deadlock. Deliberate acceptance, as defined here, instead ignores a sufficiently rare problem rather than actively applying those controls. It is still a possible policy choice, but is the only listed one outside the active-control category specified by the revised stem.

Why each choice is right or wrong

A - Incorrect: Avoidance actively restricts grants to preserve a safe state.

B - Incorrect: Detection and recovery actively identifies and resolves deadlock.

C - Incorrect: Prevention actively removes a necessary deadlock condition.

D - Correct: Intentional ignoring is a policy choice but does not actively apply the specified controls.

Primary note proof

Study-note section OS26

> Detection asks whether the current state is deadlocked. Recovery can terminate selected processes or reclaim safely preemptable resources and roll affected work back to a checkpoint. Termination and rollback are both possible recovery tools; choose according to the resource and process conditions.

Study-note section OS26

> Ignoring a sufficiently rare problem is sometimes called the ostrich policy, but real systems still prevent or detect selected deadlocks.

Study-note section OS24

> Prevention constrains how requests are made.

Study-note section OS25

Avoidance allows requests only if the resulting state remains safe.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

- Loads a program into memory.
- Executes the program.
- Handles program's execution.
- Provides a mechanism for process synchronization.
- Provides a mechanism for process communication.
- Provides a mechanism for deadlock handling.

Question sources

Computer Science MCQ practice collection · Question 680

Question 154 | AM0681

In a resource-allocation graph with one instance of each resource type, which listed roles are valid? Read B as direct resource-to-resource edges and D as showing permission to forcibly withdraw resources. Select all that apply.

A. To represent the allocation of resources to processes.

B. To represent the dependencies between resources.

C. To represent the circular wait condition.

D. To represent the preemption of resources.

Hint

Follow request and assignment arrows and check how many instances each resource has.

Answer A, C: To represent the allocation of resources to processes.; To represent the circular wait condition.

An assignment edge records which process holds a resource instance, and a request edge records what a process is waiting for. Following alternating edges can form a circular wait. In the stated single-instance model a cycle establishes deadlock. The graph does not use direct resource-to-resource dependency edges and does not encode a general permission to preempt resources, so A and C are the valid listed roles.

Why each choice is right or wrong

A - Correct: Assignment edges represent resource holdings by processes.

B - Incorrect: Standard graph edges connect processes and resources, not resources directly to other resources.

C - Correct: A cycle of request and assignment edges represents circular wait.

D - Incorrect: Resource-preemption permission is not encoded by the stated request/assignment edges.

Primary note proof

Study-note section OS23

> In a resource-allocation graph, P→R is a request and an instance R→P is an assignment. Follow these directions to form waits. No cycle implies no deadlock; with one instance of every resource type, a cycle implies deadlock. With multiple instances a cycle alone is insufficient: another instance may permit completion. A wait-for graph replaces single-instance resource nodes with P→Q when P waits for Q.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

> • Loads a program into memory.
> • Executes the program.
> • Handles program's execution.
> • Provides a mechanism for process synchronization.
> • Provides a mechanism for process communication.
> • Provides a mechanism for deadlock handling.

Question sources

Computer Science MCQ practice collection · Question 681

Question 155 | AM0682

Which resource allocation policy can help prevent deadlock?

A. First-Come, First-Served (FCFS)

B. Round Robin (RR)

C. Shortest Job Next (SJN)

D. Banker's Algorithm

Hint

Compute remaining needs and simulate feasible completions before deciding whether an allocation is safe.

Answer D: Banker’s algorithm, technically an avoidance policy.

OS25 states that Banker grants requests only while preserving a safe state. That can prevent entry into a deadlocked state in the broad everyday sense used by can help prevent, although its technical category is avoidance rather than condition-breaking prevention. FCFS, RR and SJN are CPU scheduling policies and do not themselves test resource-allocation safety.

Why each choice is right or wrong

A - Incorrect: FCFS orders CPU bursts by arrival rather than testing maximum resource claims and safe allocations.

B - Incorrect: RR schedules CPU turns; time slicing does not establish safe allocation of locks or other resources.

C - Incorrect: SJN selects shortest next CPU bursts rather than performing a resource-state safety test.

D - Correct: Banker admits resource requests only when the resulting state is safe; its strict classification is avoidance.

Primary note proof

Study-note section OS25

> Avoidance allows requests only if the resulting state remains safe. Banker requires maximum claims. Compute Need=Max−Allocation.

Study-note section OS14

> Non-preemptive scheduling keeps a running CPU burst until completion, blocking or voluntary yielding. Preemptive scheduling can return running work to ready, usually after a timer quantum or a more urgent arrival. FCFS and non-preemptive SJF are non-preemptive; RR and SRTF are preemptive. Priority can be either, so its name alone cannot decide. Cooperative describes relying on voluntary release.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

> • Loads a program into memory.
> • Executes the program.
> • Handles program's execution.
> • Provides a mechanism for process synchronization.
> • Provides a mechanism for process communication.
> • Provides a mechanism for deadlock handling.

Operating System module | PDF page 23 | lines 12-17

> 7.1 First Come First Serve (FCFS)
> • Jobs are executed on first come, first serve basis.
> • Easy to understand and implement.
> • Poor in performance as average wait time is high.
> Wait time of each process is following
> Process Wait Time: Service Time - Arrival Time

Operating System module | PDF page 25 | lines 10-14

> 7.4 Round Robin Scheduling
> • Each process is provided a fix time to execute called quantum.

• Once a process is executed for given time period. Process is preempted and other process executes for given time period.
• Context switching is used to save states of preempted processes.

Operating System module | PDF page 23 | lines 23-26

7.2 Shortest Job First (SJF)
• Best approach to minimize waiting time.
• Impossible to implement
• Processer should know in advance how much time process will take.

Question sources

Computer Science MCQ practice collection · Question 682

Question 156 | AM0683

What is the purpose of the Banker's Algorithm in deadlock avoidance?

A. To break the hold and wait condition.

B. To allocate resources in a fair and equitable manner.

C. To detect deadlocked processes and recover from deadlock.

D. To ensure that resource requests will not lead to an unsafe state.

Hint

Compute remaining needs and simulate feasible completions before deciding whether an allocation is safe.

Answer D: Ensure each granted request preserves a safe state.

Banker’s algorithm checks whether a tentative resource grant preserves a safe state. It computes remaining needs and simulates an order in which all processes can finish, granting only requests within the declared need and available resources that pass this safety test. It is an avoidance policy, not a fairness guarantee, an already-deadlocked-state detector, or prevention by eliminating hold-and-wait.

Why each choice is right or wrong

A - Incorrect: Banker does not require removing hold-and-wait; it checks whether tentative allocation remains safe.

B - Incorrect: Its safety criterion is not a guarantee of fair or equal allocation among processes.

C - Incorrect: Detection and recovery address actual deadlock, whereas Banker checks safety before granting a request.

D - Correct: Preserving a safe state after each grant is the purpose of the Banker avoidance test.

Primary note proof

Study-note section OS25

Avoidance allows requests only if the resulting state remains safe. Banker requires maximum claims. Compute Need=Max−Allocation. Set Work=Available and repeatedly find an unfinished process with Need≤Work componentwise; simulate completion by adding its Allocation to Work. Finishing all gives a safe sequence. Tentatively grant a request only if it is within Need and Available and the safety test succeeds. Unsafe does not mean already deadlocked. Banker is avoidance, not CPU scheduling, detection or prevention by breaking a condition.

Study-note section OS24

Prevention constrains how requests are made.

Study-note section OS26

Detection asks whether the current state is deadlocked. Recovery can terminate selected processes or reclaim safely preemptable resources and roll affected work back to a checkpoint.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

- Loads a program into memory.
- Executes the program.
- Handles program's execution.
- Provides a mechanism for process synchronization.
- Provides a mechanism for process communication.
- Provides a mechanism for deadlock handling.

Question sources

Computer Science MCQ practice collection · Question 683

Question 157 | AM0684

Which of the following is a characteristic of preemptive scheduling?

A. Tasks cannot be interrupted once started.

B. Tasks are allowed to complete their execution.

C. Tasks can be interrupted and rescheduled.

D. Tasks are scheduled based on their arrival time.

Hint

Check whether the running burst can be interrupted before completion or blocking.

Answer C: Tasks can be interrupted and rescheduled.

PDF p25 explicitly preempts a process after its quantum; p19 returns forcibly removed work to the ready queue. C states interrupting and rescheduling that work.

Why each choice is right or wrong

A - Incorrect: Contradicts the explicit forced removal/preemption on pp19,25.

B - Incorrect: Tasks can eventually complete, but permission to finish does not distinguish preemptive scheduling.

C - Correct: Matches the interruption/rescheduling mechanism.

D - Incorrect: Arrival ordering defines FCFS (p23), not preemption.

Primary note proof

Study-note section OS14

> Non-preemptive scheduling keeps a running CPU burst until completion, blocking or voluntary yielding. Preemptive scheduling can return running work to ready, usually after a timer quantum or a more urgent arrival. FCFS and non-preemptive SJF are non-preemptive; RR and SRTF are preemptive. Priority can be either, so its name alone cannot decide. Cooperative describes relying on voluntary release.

Study-note section OS15

> FCFS runs ready bursts in arrival order and ordinarily does not preempt.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 25 | lines 10-14

> 7.4 Round Robin Scheduling
> • Each process is provided a fix time to execute called quantum.
> • Once a process is executed for given time period. Process is preempted and other process executes for given time period.
> • Context switching is used to save states of preempted processes.

Operating System module | PDF page 19 | lines 6-12

> A newly arrived process is put in the ready queue. Processes waits in ready queue for allocating the CPU. Once the CPU is assigned to a process, then that process will execute. While executing the process, any one of the following events can occur.
> • The process could issue an I/O request and then it would be placed in an I/O queue.
> • The process could create new sub process and will wait for its termination.
> • The process could be removed forcibly from the CPU, as a result of interrupt and put back in the ready queue.

Question sources

Computer Science MCQ practice collection · Question 684

Question 158 | AM0685

Which listed scheduling policies are non-preemptive? SJN means non-preemptive shortest-next-burst scheduling; D specifically uses immediate preemption for a higher-priority arrival. Select all that apply.

A. Round Robin

B. First-Come, First-Served (FCFS)

C. Shortest Job Next (SJN)

D. Preemptive Priority Scheduling

Hint

Check whether the running burst can be interrupted before completion or blocking.

Answer B, C: First-Come, First-Served (FCFS); Shortest Job Next (SJN)

FCFS ordinarily serves each ready burst in arrival order without forcibly taking the CPU away. SJN, as explicitly defined, chooses the shortest ready burst and then lets it continue non-preemptively. RR can interrupt a burst at its quantum boundary, while the stated priority variant interrupts for a higher-priority arrival. Thus FCFS and SJN are the two non-preemptive policies in this list.

Why each choice is right or wrong

A - Incorrect: RR uses quantum expiry to preempt unfinished work.

B - Correct: FCFS ordinarily lets the selected burst continue without forced preemption.

C - Correct: SJN is explicitly the non-preemptive shortest-next-burst policy.

D - Incorrect: This option explicitly preempts for a higher-priority arrival.

Primary note proof

Study-note section OS14

> Non-preemptive scheduling keeps a running CPU burst until completion, blocking or voluntary yielding. Preemptive scheduling can return running work to ready, usually after a timer quantum or a more urgent arrival. FCFS and non-preemptive SJF are non-preemptive; RR and SRTF are preemptive. Priority can be either, so its name alone cannot decide. Cooperative describes relying on voluntary release.

Study-note section OS17

> Priority scheduling chooses the highest-priority ready job. Check whether a smaller or larger numeric value means higher priority. State whether a high-priority arrival preempts.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 23 | lines 12-17

7.1 First Come First Serve (FCFS)
• Jobs are executed on first come, first serve basis.
• Easy to understand and implement.
• Poor in performance as average wait time is high.
Wait time of each process is following
Process Wait Time: Service Time - Arrival Time

Operating System module | PDF page 23 | lines 23-26

7.2 Shortest Job First (SJF)
• Best approach to minimize waiting time.
• Impossible to implement
• Processer should know in advance how much time process will take.

Operating System module | PDF page 24 | lines 10-15

7.3 Priority Based Scheduling
• Each process is assigned a priority. Process with highest priority is to be executed first and so on.
• Processes with same priority are executed on first come first serve basis.
• Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Operating System module | PDF page 25 | lines 10-14

7.4 Round Robin Scheduling
• Each process is provided a fix time to execute called quantum.
• Once a process is executed for given time period. Process is preempted and other process executes for given time period.
• Context switching is used to save states of preempted processes.

Question sources

Computer Science MCQ practice collection · Question 685

Question 159 | AM0686

Which scheduling algorithm is based on the priority assigned to each task?

A. Round Robin

B. First-Come, First-Served (FCFS)

C. Shortest Job Next (SJN)

D. Priority Scheduling

Hint

Check priority direction, preemption and whether waiting can change effective rank.

Answer D: Priority Scheduling

PDF p24 assigns a priority to each process and executes the highest priority first. That is D; other policies use quantum, arrival or execution length.

Why each choice is right or wrong

A - Incorrect: RR uses fixed quantum (p25).

B - Incorrect: FCFS follows arrival order (p23).

C - Incorrect: SJN uses job length (p23).

D - Correct: Assigned priority is the explicit selection rule (p24).

Primary note proof

Study-note section OS17

> Priority scheduling chooses the highest-priority ready job.

Study-note section OS18

> RR serves a ready queue cyclically for at most quantum q per turn.

Study-note section OS15

> FCFS runs ready bursts in arrival order and ordinarily does not preempt.

Study-note section OS16

> Non-preemptive SJF/SJN chooses the shortest next burst among currently ready jobs.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 24 | lines 10-15

> 7.3 Priority Based Scheduling
> • Each process is assigned a priority. Process with highest priority is to be executed first and so on.
> • Processes with same priority are executed on first come first serve basis.
> • Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Operating System module | PDF page 25 | lines 10-14

> 7.4 Round Robin Scheduling
> • Each process is provided a fix time to execute called quantum.
> • Once a process is executed for given time period. Process is preempted and other process executes for given time period.
> • Context switching is used to save states of preempted processes.

Operating System module | PDF page 23 | lines 12-17

> 7.1 First Come First Serve (FCFS)
> • Jobs are executed on first come, first serve basis.

• Easy to understand and implement.
• Poor in performance as average wait time is high.
Wait time of each process is following
Process Wait Time: Service Time - Arrival Time

Operating System module | PDF page 23 | lines 23-26

7.2 Shortest Job First (SJF)
• Best approach to minimize waiting time.
• Impossible to implement
• Processer should know in advance how much time process will take.

Question sources

Computer Science MCQ practice collection · Question 686

Computer Science MCQ practice collection · Question 697

Question 160 | AM0687

User A has four continuously ready tasks and user B has one. The policy goal is equal aggregate CPU entitlement for each user, independent of task count. Which listed policy directly implements that goal when configured with equal per-user weights?

A. Round Robin

B. First-Come, First-Served (FCFS)

C. Shortest Job Next (SJN)

D. Fair-Share Scheduling

Hint

Identify the allocation entity, entitlement weights and whether the guarantee is probabilistic.

Answer D: Fair-Share Scheduling

Equal-turn RR gives each of the five tasks roughly one fifth of CPU service, so user A receives about 80 percent and B about 20 percent. Equal per-user fair-share instead gives each user half, dividing A’s half among its four tasks. FCFS and SJN do not enforce that user entitlement. The explicitly defined participant scope makes fair-share the unique matching policy.

Why each choice is right or wrong

A - Incorrect: Per-task RR gives A four fifths of service rather than equal per-user shares.

B - Incorrect: FCFS orders bursts by arrival and does not enforce per-user entitlements.

C - Incorrect: SJN favors short bursts and does not allocate equal user shares.

D - Correct: Equal per-user fair-share weights directly implement the stated half-and-half goal.

Primary note proof

Study-note section OS20

Fair-share scheduling allocates according to stated weights, potentially per user/group rather than per process. Equal per-process RR can let a user obtain more aggregate CPU by launching more processes. Proportional shares address that policy choice; fair need not mean equal.

Study-note section OS20

If user A runs four ready tasks and B runs one, plain RR gives A about 80% overall. A policy granting each user half the CPU instead gives A's tasks about 12.5% each and B's task 50%.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 25 | lines 10-14

7.4 Round Robin Scheduling
• Each process is provided a fix time to execute called quantum.
• Once a process is executed for given time period. Process is preempted and other process executes for given time period.
• Context switching is used to save states of preempted processes.

Operating System module | PDF page 23 | lines 12-17

7.1 First Come First Serve (FCFS)
• Jobs are executed on first come, first serve basis.
• Easy to understand and implement.
• Poor in performance as average wait time is high.
Wait time of each process is following
Process Wait Time: Service Time - Arrival Time

Operating System module | PDF page 23 | lines 23-26

7.2 Shortest Job First (SJF)
• Best approach to minimize waiting time.
• Impossible to implement
• Processer should know in advance how much time process will take.

Question sources

Computer Science MCQ practice collection · Question 687

Question 161 | AM0688

A ready target has finitely many jobs ahead, all CPU bursts and RR quanta are finite and positive, context overhead is finite, and later arrivals join the tail without bypassing existing peers. Infinitely many later short or high-priority jobs may arrive over time. Which policies prevent that target from being postponed forever by later arrivals? D uses fixed priorities without aging. Select all that apply.

A. Round Robin

B. First-Come, First-Served (FCFS)

C. Shortest Job Next (SJN)

D. Fixed-priority scheduling without aging

Hint

Follow arrival order, include idle periods and distinguish waiting from completion time.

Answer A, B: Round Robin; First-Come, First-Served (FCFS)

RR gives the target a turn after the finite work ahead has received bounded turns. FCFS serves the finite earlier bursts before later arrivals, so the target eventually starts. SJN can repeatedly prefer later short jobs, and fixed priority can repeatedly prefer later high-priority jobs. The assumptions distinguish guaranteed eventual service from policies that permit an endless stream to bypass the target.

Why each choice is right or wrong

A - Correct: RR's finite cyclic turns reach the target without later-arrival bypass.

B - Correct: FCFS drains the finite predecessor work before serving later arrivals.

C - Incorrect: An endless sequence of short arrivals can keep a long target waiting under SJN.

D - Incorrect: An endless sequence of higher-priority arrivals can starve a low-priority target without aging.

Primary note proof

Study-note section OS15

> FCFS runs ready bursts in arrival order and ordinarily does not preempt. If the CPU is idle, jump to the next arrival. Long CPU-bound work can make many short jobs wait: the convoy effect. A finite burst with finitely many jobs ahead eventually starts, but waiting can still be large.

Study-note section OS18

> RR serves a ready queue cyclically for at most quantum q per turn. Unfinished jobs return to the tail; completed/blocked jobs leave.

Study-note section OS17

> Fixed-priority scheduling can starve low-priority jobs under continued high-priority load. SJF can also starve long jobs; starvation is not exclusive to fixed-priority scheduling.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 25 | lines 10-14

7.4 Round Robin Scheduling
• Each process is provided a fix time to execute called quantum.
• Once a process is executed for given time period. Process is preempted and other process executes for given time period.
• Context switching is used to save states of preempted processes.

Operating System module | PDF page 23 | lines 12-17

7.1 First Come First Serve (FCFS)
• Jobs are executed on first come, first serve basis.
• Easy to understand and implement.
• Poor in performance as average wait time is high.
Wait time of each process is following
Process Wait Time: Service Time - Arrival Time

Operating System module | PDF page 23 | lines 23-26

7.2 Shortest Job First (SJF)
• Best approach to minimize waiting time.
• Impossible to implement
• Processer should know in advance how much time process will take.

Question sources

Computer Science MCQ practice collection · Question 688

Question 162 | AM0689

Which scheduling algorithm is based on the estimated remaining time required for each task?

A. Round Robin

B. First-Come, First-Served (FCFS)

C. Shortest Job Next (SJN)

D. Shortest Remaining Time First (SRTF)

Hint

Compare the ready jobs using the stated burst or remaining-time rule.

Answer D: Shortest Remaining Time First (SRTF).

OS16 says SRTF reconsiders on arrivals and selects the shortest remaining work. This matches the estimated remaining-time criterion in the stem. SJN chooses the shortest next burst non-preemptively, while RR uses cyclic quanta and FCFS arrival order. The distinction matters after execution has begun: the originally shortest burst need not be the currently shortest remaining job.

Why each choice is right or wrong

A - Incorrect: RR rotates bounded turns rather than selecting the job with the smallest estimated remaining execution.

B - Incorrect: FCFS follows arrival order rather than comparing estimated remaining CPU work.

C - Incorrect: Non-preemptive SJN compares next burst lengths when selecting, not continuously reconsidered remaining work.

D - Correct: SRTF explicitly selects the shortest remaining work and can preempt on a shorter arrival.

Primary note proof

Study-note section OS16

> Non-preemptive SJF/SJN chooses the shortest next burst among currently ready jobs. SRTF reconsiders on arrivals and selects the shortest remaining work; a shorter newcomer can preempt. Future burst lengths normally need estimates.

Study-note section OS16

> Repeated short arrivals can starve a long job. The shortest total original burst is not necessarily the shortest remaining burst after execution has begun.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 23 | lines 23-26

> 7.2 Shortest Job First (SJF)
> • Best approach to minimize waiting time.
> • Impossible to implement
> • Processer should know in advance how much time process will take.

Operating System module | PDF page 25 | lines 10-14

> 7.4 Round Robin Scheduling
> • Each process is provided a fix time to execute called quantum.
> • Once a process is executed for given time period. Process is preempted and other process executes for given time period.
> • Context switching is used to save states of preempted processes.

Operating System module | PDF page 24 | lines 10-15

7.3 Priority Based Scheduling
• Each process is assigned a priority. Process with highest priority is to be executed first and so on.
• Processes with same priority are executed on first come first serve basis.
• Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Question sources

Computer Science MCQ practice collection · Question 689

Question 163 | AM0690

Which policies can starve a ready job through repeated preference for later arrivals? Assume correct implementations, finite bursts, ordinary RR with a positive quantum, FCFS without bypass, non-preemptive SJN, and fixed-priority scheduling without aging. Select all that apply.

A. Round Robin

B. First-Come, First-Served (FCFS)

C. Shortest Job Next (SJN)

D. Priority Scheduling

Hint

Check priority direction, preemption and whether waiting can change effective rank.

Answer C, D: Shortest Job Next (SJN); Priority Scheduling

SJN can continually choose newly arrived short bursts while an older long job remains ready. Fixed priority without aging can likewise keep choosing high-priority arrivals over a low-priority target. Ordinary RR cycles peers, and FCFS does not let later arrivals overtake a finite set of predecessors. Thus the preference rules of SJN and fixed priority create the two stated starvation risks.

Why each choice is right or wrong

A - Incorrect: Ordinary RR returns ready jobs to recurring turns rather than repeatedly bypassing a peer.

B - Incorrect: FCFS preserves arrival order, so later arrivals do not overtake the target.

C - Correct: Repeated short arrivals can starve an older long SJN job.

D - Correct: Continued high-priority arrivals can starve a low-priority job without aging.

Primary note proof

Study-note section OS17

Fixed-priority scheduling can starve low-priority jobs under continued high-priority load. SJF can also starve long jobs; starvation is not exclusive to fixed-priority scheduling.

Study-note section OS15

FCFS runs ready bursts in arrival order and ordinarily does not preempt. If the CPU is idle, jump to the next arrival. Long CPU-bound work can make many short jobs wait: the convoy effect. A finite burst with finitely many jobs ahead eventually starts, but waiting can still be large.

Study-note section OS18

RR serves a ready queue cyclically for at most quantum q per turn. Unfinished jobs return to the tail; completed/blocked jobs leave.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 23 | lines 23-26

7.2 Shortest Job First (SJF)
• Best approach to minimize waiting time.
• Impossible to implement
• Processer should know in advance how much time process will take.

Operating System module | PDF page 24 | lines 10-15

7.3 Priority Based Scheduling
• Each process is assigned a priority. Process with highest priority is to be executed first and so on.
• Processes with same priority are executed on first come first serve basis.
• Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Operating System module | PDF page 25 | lines 10-14

7.4 Round Robin Scheduling
• Each process is provided a fix time to execute called quantum.
• Once a process is executed for given time period. Process is preempted and other process executes for given time period.
• Context switching is used to save states of preempted processes.

Question sources

Computer Science MCQ practice collection · Question 690

Question 164 | AM0691

Which scheduling algorithm uses time slices to allocate CPU time to tasks?

A. Round Robin

B. First-Come, First-Served (FCFS)

C. Shortest Job Next (SJN)

D. Priority Scheduling

Hint

Trace bounded turns in ready-queue order and include the stated overhead and arrivals.

Answer A: Round Robin

Each RR process receives a fixed time quantum (PDF p25); it is preempted when that period expires. A is the direct time-slice mechanism.

Why each choice is right or wrong

A - Correct: Exactly the fixed-quantum rule (p25).

B - Incorrect: Arrival-order service does not itself define repeated time slices (p23).

C - Incorrect: SJN bases selection on execution duration (p23).

D - Incorrect: Priority bases selection on assigned rank (p24); it need not use a quantum.

Primary note proof

Study-note section OS18

> RR serves a ready queue cyclically for at most quantum q per turn. Unfinished jobs return to the tail; completed/blocked jobs leave.

Study-note section OS15

> FCFS runs ready bursts in arrival order and ordinarily does not preempt.

Study-note section OS16

> Non-preemptive SJF/SJN chooses the shortest next burst among currently ready jobs.

Study-note section OS17

> Priority scheduling chooses the highest-priority ready job.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 25 | lines 10-14

> 7.4 Round Robin Scheduling
> • Each process is provided a fix time to execute called quantum.
> • Once a process is executed for given time period. Process is preempted and other process executes for given time period.
> • Context switching is used to save states of preempted processes.

Operating System module | PDF page 23 | lines 12-17

> 7.1 First Come First Serve (FCFS)
> • Jobs are executed on first come, first serve basis.
> • Easy to understand and implement.
> • Poor in performance as average wait time is high.
> Wait time of each process is following
> Process Wait Time: Service Time - Arrival Time

Operating System module | PDF page 23 | lines 23-26

7.2 Shortest Job First (SJF)
• Best approach to minimize waiting time.
• Impossible to implement
• Processer should know in advance how much time process will take.

Operating System module | PDF page 24 | lines 10-15

7.3 Priority Based Scheduling
• Each process is assigned a priority. Process with highest priority is to be executed first and so on.
• Processes with same priority are executed on first come first serve basis.
• Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Question sources

Computer Science MCQ practice collection · Question 691

Question 165 | AM0692

For a fixed finite set of simultaneous ready jobs with known CPU bursts, no I/O and zero switch cost, which listed non-preemptive policy minimizes average waiting over all possible execution orders for arbitrary burst lengths?

A. Round Robin

B. First-Come, First-Served (FCFS)

C. Shortest Job Next (SJN)

D. Shortest Remaining Time First (SRTF)

Hint

Compare the ready jobs using the stated burst or remaining-time rule.

Answer C: Shortest Job Next (SJN)

For the simultaneous known-burst workload, shortest-job order minimizes total waiting. Consider adjacent bursts L>S after an elapsed prefix T. Long then short gives waits T and T+L; short then long gives T and T+S, reducing their sum by L−S. Jobs outside the pair are unaffected. Repeatedly removing inversions gives SJN order. FCFS does not enforce it, while RR and SRTF are preemptive policies.

Why each choice is right or wrong

A - Incorrect: RR is preemptive and is outside the requested non-preemptive policy class.

B - Incorrect: FCFS can tie on favorable orders but does not minimize waiting for arbitrary burst orders.

C - Correct: SJN selects the shortest ready bursts and gives the standard minimum-average-waiting order.

D - Incorrect: SRTF is the preemptive remaining-time policy rather than the requested non-preemptive one.

Primary note proof

Study-note section OS16

> Non-preemptive SJF/SJN chooses the shortest next burst among currently ready jobs. SRTF reconsiders on arrivals and selects the shortest remaining work; a shorter newcomer can preempt. Future burst lengths normally need estimates. SJF minimizes average waiting for the usual fixed simultaneous ready set with known bursts and no preemption/switch cost; that theorem is not a workload-independent comparison against every preemptive policy.

Study-note section OS13

> For arrival A, first start S and completion C, response=S−A and turnaround=C−A. For a single CPU burst B with no I/O, waiting=C−A−B.

Study-note section OS14

> FCFS and non-preemptive SJF are non-preemptive; RR and SRTF are preemptive.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 23 | lines 23-26

> 7.2 Shortest Job First (SJF)
> • Best approach to minimize waiting time.
> • Impossible to implement
> • Processer should know in advance how much time process will take.

Operating System module | PDF page 25 | lines 10-14

> 7.4 Round Robin Scheduling
> • Each process is provided a fix time to execute called quantum.
> • Once a process is executed for given time period. Process is preempted and other process executes for given time period.
> • Context switching is used to save states of preempted processes.

Operating System module | PDF page 23 | lines 12-17

> 7.1 First Come First Serve (FCFS)
> • Jobs are executed on first come, first serve basis.
> • Easy to understand and implement.
> • Poor in performance as average wait time is high.
> Wait time of each process is following
> Process Wait Time: Service Time - Arrival Time

Question sources

Computer Science MCQ practice collection · Question 692

Question 166 | AM0693

Which scheduling algorithm provides better response time for interactive tasks?

A. Round Robin

B. First-Come, First-Served (FCFS)

C. Shortest Job Next (SJN)

D. Priority Scheduling

Hint

Trace bounded turns in ready-queue order and include the stated overhead and arrivals.

Answer A: Round Robin

Time-sharing seeks short response by giving users short CPU bursts (PDF p7). RR implements fixed quantum turns (p25), so A is a well-supported interactive choice with a suitably short quantum.

Why each choice is right or wrong

A - Correct: Repeated short quanta implement the p7 interactive strategy.

B - Incorrect: Arrival-order long work can delay later interactive work (p23).

C - Incorrect: SJN needs duration estimates (p23), not a guaranteed prompt periodic turn.

D - Incorrect: Priority may help selected interactive jobs, but depends on assigned ranks rather than the general time-sharing mechanism.

Primary note proof

Study-note section OS18

> RR serves a ready queue cyclically for at most quantum q per turn. Unfinished jobs return to the tail; completed/blocked jobs leave. Specify how arrivals at a quantum boundary enter the queue. A large quantum approaches FCFS; tiny quanta increase switch overhead. RR gives recurring opportunities, not identical total CPU consumption for jobs of different lengths.

Study-note section OS21

> RR is a conventional interactive choice because turns improve responsiveness, subject to quantum and overhead.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 25 | lines 10-14

> 7.4 Round Robin Scheduling
> • Each process is provided a fix time to execute called quantum.
> • Once a process is executed for given time period. Process is preempted and other process executes for given time period.
> • Context switching is used to save states of preempted processes.

Operating System module | PDF page 7 | lines 20-33

Time sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time. Time-sharing or multitasking is a logical extension of multiprogramming. Processor's time which is shared among multiple users simultaneously is termed as time-sharing. The main difference between Multi-programmed Batch Systems and Time-Sharing Systems is that in case of multi-programmed batch systems, objective is to maximize processor use, whereas in Time-Sharing Systems objective is to minimize response time.
Multiple jobs are executed by the CPU by switching between them, but the switches occur so frequently. Thus, the user can receive an immediate response. For example, in a transaction processing, processor execute each user program in a short burst or quantum of computation. That is if n users are present, each user can get time quantum. When the user submits the command, the response time is in few seconds at most.
Operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time. Computer systems that were designed primarily as batch systems

Operating System module | PDF page 23 | lines 12-17

7.1 First Come First Serve (FCFS)
• Jobs are executed on first come, first serve basis.
• Easy to understand and implement.
• Poor in performance as average wait time is high.
Wait time of each process is following
Process Wait Time: Service Time - Arrival Time

Operating System module | PDF page 23 | lines 23-26

7.2 Shortest Job First (SJF)
• Best approach to minimize waiting time.
• Impossible to implement
• Processer should know in advance how much time process will take.

Operating System module | PDF page 24 | lines 10-15

7.3 Priority Based Scheduling
• Each process is assigned a priority. Process with highest priority is to be executed first and so on.
• Processes with same priority are executed on first come first serve basis.
• Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Question sources

Computer Science MCQ practice collection · Question 693

Computer Science MCQ practice collection · Question 700

Question 167 | AM0694

Which listed policy is preemptive? SJN denotes non-preemptive shortest-job-next, and D specifically keeps the running burst until completion, blocking or voluntary release.

A. First-Come, First-Served (FCFS)

B. Round Robin

C. Shortest Job Next (SJN)

D. Non-preemptive Priority Scheduling

Hint

Check whether the running burst can be interrupted before completion or blocking.

Answer B: Round Robin

RR limits a turn to a quantum and can return an unfinished runnable job to the ready queue. That is preemption. FCFS and the stated SJN variant keep a selected burst until it ends or blocks. The priority option is explicitly non-preemptive, so a higher-priority arrival does not interrupt its current burst. RR is therefore the unique preemptive policy in this list.

Why each choice is right or wrong

A - Incorrect: FCFS ordinarily does not forcibly interrupt the selected burst.

B - Correct: RR preempts unfinished work at quantum expiry.

C - Incorrect: SJN is explicitly the non-preemptive shortest-job policy.

D - Incorrect: The stated priority variant is explicitly non-preemptive.

Primary note proof

Study-note section OS14

> Non-preemptive scheduling keeps a running CPU burst until completion, blocking or voluntary yielding. Preemptive scheduling can return running work to ready, usually after a timer quantum or a more urgent arrival. FCFS and non-preemptive SJF are non-preemptive; RR and SRTF are preemptive. Priority can be either, so its name alone cannot decide. Cooperative describes relying on voluntary release.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 25 | lines 10-14

> 7.4 Round Robin Scheduling
> • Each process is provided a fix time to execute called quantum.
> • Once a process is executed for given time period. Process is preempted and other process executes for given time period.
> • Context switching is used to save states of preempted processes.

Operating System module | PDF page 23 | lines 12-17

7.1 First Come First Serve (FCFS)
• Jobs are executed on first come, first serve basis.
• Easy to understand and implement.
• Poor in performance as average wait time is high.
Wait time of each process is following
Process Wait Time: Service Time - Arrival Time

Operating System module | PDF page 23 | lines 23-26

7.2 Shortest Job First (SJF)
• Best approach to minimize waiting time.
• Impossible to implement
• Processer should know in advance how much time process will take.

Operating System module | PDF page 24 | lines 10-15

7.3 Priority Based Scheduling
• Each process is assigned a priority. Process with highest priority is to be executed first and so on.
• Processes with same priority are executed on first come first serve basis.
• Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Question sources

Computer Science MCQ practice collection · Question 694

Question 168 | AM0695

Which listed policy allocates CPU according to declared user/group entitlement weights, so fairness can mean unequal proportional shares rather than arrival order?

A. First-Come, First-Served (FCFS)

B. Shortest Job Next (SJN)

C. Priority Scheduling

D. Fair-Share Scheduling

Hint

Identify the allocation entity, entitlement weights and whether the guarantee is probabilistic.

Answer D: Fair-Share Scheduling

Fair-share scheduling allocates service according to stated weights, which may apply to users or groups rather than individual processes. That directly implements proportional entitlement. FCFS respects arrival order, SJN chooses shorter bursts, and priority chooses a ranking; those rules do not themselves specify weighted user/group shares. The revised fairness criterion therefore identifies fair-share without implying every fair policy must use equal shares.

Why each choice is right or wrong

A - Incorrect: FCFS provides arrival ordering rather than declared weighted entitlements.

B - Incorrect: SJN selects by burst length rather than entitlement weights.

C - Incorrect: Priority ranking does not itself implement proportional user/group shares.

D - Correct: Fair-share scheduling allocates according to declared weights and participant scope.

Primary note proof

Study-note section OS20

> Fair-share scheduling allocates according to stated weights, potentially per user/group rather than per process. Equal per-process RR can let a user obtain more aggregate CPU by launching more processes. Proportional shares address that policy choice; fair need not mean equal.

Study-note section OS15

> FCFS runs ready bursts in arrival order and ordinarily does not preempt.

Study-note section OS16

> Non-preemptive SJF/SJN chooses the shortest next burst among currently ready jobs.

Study-note section OS17

> Priority scheduling chooses the highest-priority ready job.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 23 | lines 12-17

> 7.1 First Come First Serve (FCFS)
> • Jobs are executed on first come, first serve basis.
> • Easy to understand and implement.
> • Poor in performance as average wait time is high.
> Wait time of each process is following
> Process Wait Time: Service Time - Arrival Time

Operating System module | PDF page 23 | lines 23-26

> 7.2 Shortest Job First (SJF)
> • Best approach to minimize waiting time.
> • Impossible to implement
> • Processer should know in advance how much time process will take.

Operating System module | PDF page 24 | lines 10-15

> 7.3 Priority Based Scheduling
> • Each process is assigned a priority. Process with highest priority is to be executed first and so on.
> • Processes with same priority are executed on first come first serve basis.

• Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Operating System module | PDF page 25 | lines 10-14

7.4 Round Robin Scheduling
• Each process is provided a fix time to execute called quantum.
• Once a process is executed for given time period. Process is preempted and other process executes for given time period.
• Context switching is used to save states of preempted processes.

Question sources

Computer Science MCQ practice collection · Question 695

Question 169 | AM0696

Which algorithm gives priority to the tasks with the shortest burst time?

A. First-Come, First-Served (FCFS)

B. Round Robin

C. Shortest Job Next (SJN)

D. Priority Scheduling

Hint

Compare the ready jobs using the stated burst or remaining-time rule.

Answer C: Shortest Job Next (SJN)

SJF on PDF p23 requires knowing how long a process will take and favors shortest jobs. C uses the synonymous SJN label. Priority can incorporate time requirements, but does not necessarily rank by shortest burst.

Why each choice is right or wrong

A - Incorrect: FCFS chooses order of arrival (p23).

B - Incorrect: RR chooses fixed turns (p25).

C - Correct: Shortest-job selection uses the known execution duration (p23).

D - Incorrect: Assigned priorities can reflect many requirements, not necessarily shortest burst (p24).

Primary note proof

Study-note section OS16

Non-preemptive SJF/SJN chooses the shortest next burst among currently ready jobs.

Study-note section OS15

FCFS runs ready bursts in arrival order and ordinarily does not preempt.

Study-note section OS18

RR serves a ready queue cyclically for at most quantum q per turn.

Study-note section OS17

Priority scheduling chooses the highest-priority ready job.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 23 | lines 23-26

7.2 Shortest Job First (SJF)
• Best approach to minimize waiting time.
• Impossible to implement
• Processer should know in advance how much time process will take.

Operating System module | PDF page 23 | lines 12-17

7.1 First Come First Serve (FCFS)
• Jobs are executed on first come, first serve basis.
• Easy to understand and implement.
• Poor in performance as average wait time is high.
Wait time of each process is following
Process Wait Time: Service Time - Arrival Time

Operating System module | PDF page 25 | lines 10-14

7.4 Round Robin Scheduling
• Each process is provided a fix time to execute called quantum.
• Once a process is executed for given time period. Process is preempted and other process executes for given time period.
• Context switching is used to save states of preempted processes.

Operating System module | PDF page 24 | lines 10-15

7.3 Priority Based Scheduling
• Each process is assigned a priority. Process with highest priority is to be executed first and so on.
• Processes with same priority are executed on first come first serve basis.
• Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Question sources

Computer Science MCQ practice collection · Question 696

Question 170 | AM0698

Which scheduling algorithm is most suitable for time-sharing systems?

A. First-Come, First-Served (FCFS)

B. Round Robin

C. Shortest Job Next (SJN)

D. Priority Scheduling

Hint

Compare interaction, CPU sharing and deadline requirements under the stated system model.

Answer B: Round Robin

PDF p7 describes time-sharing as short bursts/quantums for many users, seeking fast response. RR's fixed quantum and preemption (p25) directly implement that pattern, making B the matching policy.

Why each choice is right or wrong

A - Incorrect: A long early FCFS job can delay other users, unlike repeated short turns.

B - Correct: Fixed quantum sharing directly implements p7's time-sharing pattern.

C - Incorrect: SJN requires knowing lengths, not a periodic-turn guarantee.

D - Incorrect: Priority ordering alone does not guarantee each user a quantum.

Primary note proof

Study-note section OS02

> Time-sharing uses rapid scheduling/preemption to support interactive users.

Study-note section OS18

> RR serves a ready queue cyclically for at most quantum q per turn. Unfinished jobs return to the tail; completed/blocked jobs leave.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 7 | lines 20-33

> Time sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time. Time-sharing or multitasking is a logical extension of multiprogramming. Processor's time which is shared among multiple users simultaneously is termed as time-sharing. The main difference between Multi-programmed Batch Systems and Time-Sharing Systems is that in case of multi-programmed batch systems, objective is to maximize processor use, whereas in Time-Sharing Systems objective is to minimize response time.
> Multiple jobs are executed by the CPU by switching between them, but the switches occur so frequently. Thus, the user can receive an immediate response. For example, in a transaction

processing, processor execute each user program in a short burst or quantum of computation. That is if n users are present, each user can get time quantum. When the user submits the command, the response time is in few seconds at most.
Operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time. Computer systems that were designed primarily as batch systems

Operating System module | PDF page 25 | lines 10-14

7.4 Round Robin Scheduling
• Each process is provided a fix time to execute called quantum.
• Once a process is executed for given time period. Process is preempted and other process executes for given time period.
• Context switching is used to save states of preempted processes.

Operating System module | PDF page 23 | lines 12-17

7.1 First Come First Serve (FCFS)
• Jobs are executed on first come, first serve basis.
• Easy to understand and implement.
• Poor in performance as average wait time is high.
Wait time of each process is following
Process Wait Time: Service Time - Arrival Time

Operating System module | PDF page 23 | lines 23-26

7.2 Shortest Job First (SJF)
• Best approach to minimize waiting time.
• Impossible to implement
• Processer should know in advance how much time process will take.

Operating System module | PDF page 24 | lines 10-15

7.3 Priority Based Scheduling
• Each process is assigned a priority. Process with highest priority is to be executed first and so on.
• Processes with same priority are executed on first come first serve basis.
• Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Question sources

Computer Science MCQ practice collection · Question 698

Question 171 | AM0699

Which correctly implemented policies can still cause starvation under an ongoing stream of preferred arrivals? Assume finite bursts, ordinary RR with tail insertion, FCFS without bypass, non-preemptive SJN, and fixed priority without aging. Select all that apply.

A. First-Come, First-Served (FCFS)

B. Round Robin

C. Shortest Job Next (SJN)

D. Priority Scheduling

Hint

Check priority direction, preemption and whether waiting can change effective rank.

Answer C, D: Shortest Job Next (SJN); Priority Scheduling

Starvation can arise from the policy even when its implementation is correct. SJN repeatedly favors short arrivals over an older long burst, while fixed priority can repeatedly favor high-priority arrivals over low-priority work. FCFS preserves arrival order, and RR provides recurring turns under the stated finite-work assumptions. Thus SJN and fixed priority are the two policies with the specified inherent bypass risk.

Why each choice is right or wrong

A - Incorrect: FCFS later arrivals cannot bypass a target with finite predecessor work.

B - Incorrect: Ordinary RR returns peers to recurring turns rather than favoring newcomers indefinitely.

C - Correct: Repeated shorter arrivals can indefinitely postpone a long SJN job.

D - Correct: Continued higher-priority arrivals can indefinitely postpone low-priority work without aging.

Primary note proof

Study-note section OS17

> Fixed-priority scheduling can starve low-priority jobs under continued high-priority load. SJF can also starve long jobs; starvation is not exclusive to fixed-priority scheduling.

Study-note section OS15

> FCFS runs ready bursts in arrival order and ordinarily does not preempt. If the CPU is idle, jump to the next arrival. Long CPU-bound work can make many short jobs wait: the convoy effect. A finite burst with finitely many jobs ahead eventually starts, but waiting can still be large.

Study-note section OS18

> RR serves a ready queue cyclically for at most quantum q per turn. Unfinished jobs return to the tail; completed/blocked jobs leave.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 23 | lines 12-17

7.1 First Come First Serve (FCFS)
• Jobs are executed on first come, first serve basis.
• Easy to understand and implement.
• Poor in performance as average wait time is high.
Wait time of each process is following
Process Wait Time: Service Time - Arrival Time

Operating System module | PDF page 25 | lines 10-14

7.4 Round Robin Scheduling
• Each process is provided a fix time to execute called quantum.
• Once a process is executed for given time period. Process is preempted and other process executes for given time period.
• Context switching is used to save states of preempted processes.

Operating System module | PDF page 23 | lines 23-26

7.2 Shortest Job First (SJF)
• Best approach to minimize waiting time.
• Impossible to implement
• Processer should know in advance how much time process will take.

Operating System module | PDF page 24 | lines 10-15

7.3 Priority Based Scheduling
• Each process is assigned a priority. Process with highest priority is to be executed first and so on.
• Processes with same priority are executed on first come first serve basis.
• Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Question sources

Computer Science MCQ practice collection · Question 699

Question 172 | AM0701

Which scheduling algorithm has the potential for higher overhead due to frequent context switches?

A. First-Come, First-Served (FCFS)

B. Round Robin

C. Shortest Job Next (SJN)

D. Priority Scheduling

Hint

Trace bounded turns in ready-queue order and include the stated overhead and arrivals.

Answer B: Round Robin

Each RR quantum expiration preempts and saves state (PDF p25). PDF p21 calls context-switch time pure overhead, so frequent short quanta create the overhead risk described.

Why each choice is right or wrong

A - Incorrect: FCFS need not switch after every short quantum.

B - Correct: Repeated quantum expiration creates frequent context saves; p21 identifies their cost.

C - Incorrect: Whole-job SJN does not inherently expire a quantum repeatedly.

D - Incorrect: Priority can preempt frequently under some arrivals, but that is not inherent in its p24 ranking rule.

Primary note proof

Study-note section OS18

> A large quantum approaches FCFS; tiny quanta increase switch overhead.

Study-note section OS05

> A context switch saves one execution context and restores another; useful application work is interrupted by this overhead.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 25 | lines 10-14

> 7.4 Round Robin Scheduling
> • Each process is provided a fix time to execute called quantum.
> • Once a process is executed for given time period. Process is preempted and other process executes for given time period.
> • Context switching is used to save states of preempted processes.

Operating System module | PDF page 21 | lines 41-50

> A context switch is the mechanism to store and restore the state or context of a CPU in Process Control block so that a process execution can be resumed from the same point at a later time. Using this technique, a context switcher enables multiple processes to share a single CPU. Context switching is an essential part of a multitasking operating system features.
> When the scheduler switches the CPU from executing one process to execute another, the context switcher saves the content of all processor registers for the process being removed from the CPU, in its process descriptor. The context of a process is represented in the process control block of a process. Context switch time is pure overhead. Context switching can significantly affect performance as modern computers have a lot of general

Operating System module | PDF page 23 | lines 12-17

7.1 First Come First Serve (FCFS)
• Jobs are executed on first come, first serve basis.
• Easy to understand and implement.
• Poor in performance as average wait time is high.
Wait time of each process is following
Process Wait Time: Service Time - Arrival Time

Operating System module | PDF page 23 | lines 23-26

7.2 Shortest Job First (SJF)
• Best approach to minimize waiting time.
• Impossible to implement
• Processer should know in advance how much time process will take.

Operating System module | PDF page 24 | lines 10-15

7.3 Priority Based Scheduling
• Each process is assigned a priority. Process with highest priority is to be executed first and so on.
• Processes with same priority are executed on first come first serve basis.
• Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Question sources

Computer Science MCQ practice collection · Question 701

Question 173 | AM0702

Two continuously ready user groups have declared CPU entitlements of 3:1, independent of how many tasks each launches. Which listed policy is designed to enforce those unequal per-group shares over time?

A. First-Come, First-Served (FCFS)

B. Round Robin

C. Shortest Job Next (SJN)

D. Fair-Share Scheduling

Hint

Identify the allocation entity, entitlement weights and whether the guarantee is probabilistic.

Answer D: Fair-Share Scheduling

A fair-share policy allocates CPU according to declared weights and participant scope. Weights of 3:1 give intended group shares of 75 and 25 percent. Plain RR supplies equal turns to processes and does not implement those unequal group entitlements independently of task count. FCFS and SJN also lack that share rule. The question asks for the policy design rather than a universal finite-time deadline.

Why each choice is right or wrong

A - Incorrect: FCFS orders bursts by arrival rather than enforcing weighted group service.

B - Incorrect: Plain per-task RR does not enforce unequal group weights independent of task count.

C - Incorrect: SJN selects short bursts rather than proportional group entitlements.

D - Correct: Configured fair-share scheduling implements the declared group-weight allocation.

Primary note proof

Study-note section OS20

> Fair-share scheduling allocates according to stated weights, potentially per user/group rather than per process. Equal per-process RR can let a user obtain more aggregate CPU by launching more processes. Proportional shares address that policy choice; fair need not mean equal.

Study-note section OS18

> RR serves a ready queue cyclically for at most quantum q per turn. Unfinished jobs return to the tail; completed/blocked jobs leave.

Study-note section OS15

> FCFS runs ready bursts in arrival order and ordinarily does not preempt.

Study-note section OS16

> Non-preemptive SJF/SJN chooses the shortest next burst among currently ready jobs.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 25 | lines 10-14

> 7.4 Round Robin Scheduling
> • Each process is provided a fix time to execute called quantum.
> • Once a process is executed for given time period. Process is preempted and other process executes for given time period.
> • Context switching is used to save states of preempted processes.

Operating System module | PDF page 23 | lines 12-17

> 7.1 First Come First Serve (FCFS)
> • Jobs are executed on first come, first serve basis.
> • Easy to understand and implement.
> • Poor in performance as average wait time is high.
> Wait time of each process is following
> Process Wait Time: Service Time - Arrival Time

Operating System module | PDF page 23 | lines 23-26

> 7.2 Shortest Job First (SJF)
> • Best approach to minimize waiting time.

- Impossible to implement
- Processer should know in advance how much time process will take.

Question sources

Computer Science MCQ practice collection · Question 702

Question 174 | AM0703

Three jobs all arrive at time 0 in listed order P1, P2, P3, with CPU bursts 24, 3, 3 and priorities 2, 1, 3 (smaller number is higher). There is no I/O or overhead. Compare FCFS, RR with quantum 4, non-preemptive SJN, and non-preemptive priority; use listed order for ties. Which has the highest average turnaround for this workload?

A. First-Come, First-Served (FCFS)

B. Round Robin

C. Shortest Job Next (SJN)

D. Priority Scheduling

Hint

Construct a timeline and distinguish completion rates from individual elapsed-time measures.

Answer A: First-Come, First-Served (FCFS)

FCFS completes P1, P2, P3 at 24, 27 and 30, giving average turnaround 27. RR completes them at 30, 7 and 10, averaging 47/3. SJN completes short jobs at 3 and 6, then P1 at 30, averaging 13. Priority runs P2, P1, P3 and averages 20. FCFS is uniquely highest for this specified workload, without a universal worst-policy claim.

Why each choice is right or wrong

A - Correct: FCFS average turnaround is (24+27+30)/3=27, the largest.

B - Incorrect: RR average turnaround is (30+7+10)/3=47/3, below 27.

C - Incorrect: SJN average turnaround is (3+6+30)/3=13.

D - Incorrect: Priority average turnaround is (3+27+30)/3=20.

Primary note proof

Study-note section OS13

For arrival A, first start S and completion C, response=S−A and turnaround=C−A. For a single CPU burst B with no I/O, waiting=C−A−B.

Study-note section OS15

FCFS runs ready bursts in arrival order and ordinarily does not preempt.

Study-note section OS16

Non-preemptive SJF/SJN chooses the shortest next burst among currently ready jobs.

Study-note section OS17

Priority scheduling chooses the highest-priority ready job. Check whether a smaller or larger numeric value means higher priority. State whether a high-priority arrival preempts.

Study-note section OS18

RR serves a ready queue cyclically for at most quantum q per turn. Unfinished jobs return to the tail; completed/blocked jobs leave.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 23 | lines 12-17

7.1 First Come First Serve (FCFS)
• Jobs are executed on first come, first serve basis.
• Easy to understand and implement.
• Poor in performance as average wait time is high.
Wait time of each process is following
Process Wait Time: Service Time - Arrival Time

Operating System module | PDF page 23 | lines 23-26

7.2 Shortest Job First (SJF)
• Best approach to minimize waiting time.
• Impossible to implement
• Processer should know in advance how much time process will take.

Operating System module | PDF page 25 | lines 10-14

7.4 Round Robin Scheduling
• Each process is provided a fix time to execute called quantum.
• Once a process is executed for given time period. Process is preempted and other process executes for given time period.
• Context switching is used to save states of preempted processes.

Operating System module | PDF page 24 | lines 10-15

7.3 Priority Based Scheduling
• Each process is assigned a priority. Process with highest priority is to be executed first and so on.
• Processes with same priority are executed on first come first serve basis.
• Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Operating System module | PDF page 25 | lines 15-21

Wait time of each process is following
Process Wait Time: Service Time - Arrival Time
P0 (0-0) + (12-3) = 9

P1 (3-1) = 2
P2 6-2) + (15-9) = 10
P3 (9-3) + (18-12) = 12
Average Wait Time: (9+2+10+12) / 4 = 8.25

Operating System module | PDF page 23 | lines 16-22

Wait time of each process is following
Process Wait Time: Service Time - Arrival Time
P0 0 - 0 = 0
P1 5 - 1 = 4
P2 8 - 2 = 6
P3 16 - 3 = 13
Average Wait Time: (0+4+6+13)/4 = 5.75

Question sources

Computer Science MCQ practice collection · Question 703

Question 175 | CS0218

Which one of the following is a voluntary termination of a process?

A. Error exit

B. Killed by another process

C. Terminated by another process

D. Fatal exit

Hint

Separate creation, image replacement and termination; classify termination by its cause.

Answer A: Error exit, when the program detects an error and chooses to terminate.

OS07 explicitly classifies program-detected error exits as voluntary. Being killed or terminated by another process is imposed, and a fatal unrecoverable fault is involuntary. The distinction is the termination decision's origin, not whether the run succeeded: a program can voluntarily exit because its input is invalid or it detects another handled error.

Why each choice is right or wrong

A - Correct: A program-detected error exit is a deliberate voluntary termination path, even though execution was unsuccessful.

B - Incorrect: Being killed by another process is imposed externally and is therefore involuntary.

C - Incorrect: Termination by another process is external rather than a voluntary exit chosen by the program.

D - Incorrect: A fatal unrecoverable fault causes involuntary termination rather than a deliberate program error exit.

Primary note proof

Study-note section OS07

> Normal completion and a program-detected error exit are voluntary; an unrecoverable fault or being killed is involuntary. exit() terminates normally through the runtime; abort() requests abnormal termination.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 16 | lines 29-39

> 1. New The process is being created.
> 2.
> Ready The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run.
> 3.
> Running Process instructions are being executed (i.e. The process that is currently being executed).
> 4.
> Waiting The process is waiting for some event to occur (such as the completion of an I/O operation).
> 5. Terminated The process has finished execution.

Operating System module | PDF page 19 | lines 20-25

> Non-Running Processes that are not running are kept in queue, waiting for their turn to execute. Each entry in the queue is a pointer to a particular process. Queue is implemented by using linked list. Use of dispatcher is as follows. When a process is interrupted, that process is transferred in the waiting queue. If the process has completed or aborted, the process is discarded. In either case, the dispatcher then selects a process from the queue to execute.

Operating System module | PDF page 39 | lines 5-11

> Check an internal table for this process, to determine whether the reference was a valid or it was an invalid memory access.
> Step 2
> If the reference was invalid, terminate the process. If it was valid, but page have not yet brought in, page in the latter.
> Step 3 Find a free frame.
> Step 4 Schedule a disk operation to read the desired page into the newly allocated frame.

Question sources

Computer Science Exit Exam practice collection · Question 218

Question 176 | CS0219

Which statements are false in a system with one-to-one kernel-supported threads? In A, “children” means a process may create another process and a running thread may invoke a thread-creation operation; it does not require an OS thread-parent hierarchy. In B, “execute sequentially” means threads are required to finish one after another without interleaving. Select all that apply.

A. Both can create children

B. Threads within a process execute sequentially

C. Processes are not independent of one another

D. If one thread is blocked, another thread can run

Hint

Compare per-thread execution context with shared process resources and the stated thread mapping.

Answer B, C: Threads within a process execute sequentially; Processes are not independent of one another

Threads within one process need not finish sequentially; their execution may interleave or run on multiple CPUs. Independent processes can also exist, so C’s universal dependence claim is false. The defined creation operations make A true without imposing a thread-parent hierarchy. Under one-to-one mapping, one blocked thread need not block every other thread, making D a valid possibility rather than a false statement.

Why each choice is right or wrong

A - Incorrect: The defined process and thread creation operations are allowed, so A is true.

B - Correct: Mandatory finish-before-next sequential execution is false; threads can interleave or run in parallel.

C - Correct: Universal process dependence is false because independent processes can exist.

D - Incorrect: One-to-one mapping permits another runnable thread to execute when one blocks, so D is true.

Primary note proof

Study-note section OS08

> A thread has its own PC, registers and stack. Threads in a process share code, address space and usually open-file resources. A process can contain many threads; a thread belongs to one process at a time. Threads are not a group of processes and need not execute sequentially. Independent processes can exist; cooperating ones communicate/share data.

Study-note section OS08

> One-to-one permits another kernel thread to run when one blocks.

Study-note section OS07

Processes arise during initialization, user requests, service requests or a running process's creation call.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 27 | lines 6-14

A thread is a flow of execution through the process code, with its own program counter, system registers and stack. A thread is also called a light weight process. Threads provide a way to improve application performance through parallelism. Threads represent a software approach to improving performance of operating system by reducing the overhead thread is equivalent to a classical process.
Each thread belongs to exactly one process and no thread can exist outside a process. Each thread represents a separate flow of control. Threads have been successfully used in implementing network servers and web server. They also provide a suitable foundation for parallel execution of applications on shared memory multiprocessors. Following figure shows

Operating System module | PDF page 28 | lines 4-13

In multiple processes each process operates
independently of the others.
One thread can read, write or change
another thread's data.
8.2 Advantages of Thread
• Thread minimizes context switching time.
• Use of threads provides concurrency within a process.
• Efficient communication.
• Economy- It is more economical to create and context switch threads.
• Utilization of multiprocessor architectures to a greater scale and efficiency.

Operating System module | PDF page 29 | lines 4-16

In this case, thread management done by the Kernel. There is no thread management code in the application area. Kernel threads are supported directly by the operating system. Any application can be programmed to be multithreaded. All of the threads within an application are supported within a single process.
The Kernel maintains context information for the process as a whole and for individuals' threads within the process. Scheduling by the Kernel is done on a thread basis. The Kernel performs thread creation, scheduling and management in Kernel space. Kernel threads are generally slower to create and manage than the user threads.
Advantages
• Kernel can simultaneously schedule multiple threads from the same process on multiple processes.
• If one thread in a process is blocked, the Kernel can schedule another thread of the same process.

Operating System module | PDF page 30 | lines 4-7

Many to one model maps many user level threads to one Kernel level thread. Thread management is done in user space. When thread makes a blocking system call, the entire

process will be blocks. Only one thread can access the Kernel at a time, so multiple threads are unable to run in parallel on multiprocessors.

Operating System module | PDF page 19 | lines 6-12

A newly arrived process is put in the ready queue. Processes waits in ready queue for allocating the CPU. Once the CPU is assigned to a process, then that process will execute. While executing the process, any one of the following events can occur.
• The process could issue an I/O request and then it would be placed in an I/O queue.
• The process could create new sub process and will wait for its termination.
• The process could be removed forcibly from the CPU, as a result of interrupt and put back in the ready queue.

Operating System module | PDF page 28 | lines 20-29

In this case, application manages thread management kernel is not aware of the existence of threads. The thread library contains code for creating and destroying threads, for passing message and data between threads, for scheduling thread execution and for saving and restoring thread contexts. The application begins with a single thread and begins running in that thread.
Advantages
• Thread switching does not require Kernel mode privileges.
• User level thread can run on any operating system.
• Scheduling can be application specific in the user level thread.
• User level threads are fast to create and manage.

Question sources

Computer Science Exit Exam practice collection · Question 219

Question 177 | CS0220

Which statements are false under a classification based on simultaneous users? A single user may run multiple applications, and B means concurrent database accesses by those applications, not by additional users. In C, “at most one” describes the system’s supported capacity. Select all that apply.

A. In single user systems many users can login simultaneously

B. Single user systems can allow concurrent access to the database

C. Multiuser system can allow at most one user at a time

D. In Multiuser system many users can access the system concurrently

Hint

Compare interaction, CPU sharing and deadline requirements under the stated system model.

Answer A, C: In single user systems many users can login simultaneously; Multiuser system can allow at most one user at a time

Single-user and multiuser describe concurrent user access rather than application count. A therefore contradicts single-user status, and C contradicts the supported multiple-user capacity. One user may still run several applications accessing a database concurrently, which is the explicit meaning of B here. D states the multiuser capability correctly. The selected false statements are A and C.

Why each choice is right or wrong

A - Correct: Many simultaneous logged-in users contradict the single-user definition.

B - Incorrect: Concurrent applications belonging to one user do not violate single-user status, so B is true.

C - Correct: A capacity of at most one user contradicts the multiuser definition.

D - Incorrect: Supporting concurrent access by many users is the multiuser capability, so D is true.

Primary note proof

Study-note section OS02

> Single-user and multiuser describe concurrent user access, not the number of applications.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 54 | lines 5-17

> Portable - Portability means software can works on different types of hardware in same way. Linux kernel and application programs support their installation on any kind of hardware platform.
> Open Source - Linux source code is freely available and it is community-based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
> Multi-User - Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.
> Multiprogramming - Linux is a multiprogramming system means multiple applications can run at same time.
> Hierarchical File System - Linux provides a standard file structure in which system files/ user files are arranged.
> Shell - Linux provides a special interpreter program which can be used to execute

Operating System module | PDF page 7 | lines 20-33

> Time sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time. Time-sharing or multitasking is a logical extension of multiprogramming. Processor's time which is shared among multiple users simultaneously is termed as time-sharing. The main difference between Multi-programmed Batch Systems and Time-Sharing Systems is that in case of multi-programmed batch systems, objective is to maximize processor use, whereas in Time-Sharing Systems objective

is to minimize response time.
Multiple jobs are executed by the CPU by switching between them, but the switches occur so frequently. Thus, the user can receive an immediate response. For example, in a transaction processing, processor execute each user program in a short burst or quantum of computation. That is if n users are present, each user can get time quantum. When the user submits the command, the response time is in few seconds at most.
Operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time. Computer systems that were designed primarily as batch systems

Question sources

Computer Science Exit Exam practice collection · Question 220

Question 178 | CS0221

When a process executes code that possibly changes the value of shared data, its execution has to be mutually exclusive. Therefore, the process is in:

A. Deadlock

B. Terminated

C. Critical section

D. Blocked

Hint

Identify conflicting shared accesses and the safety or progress property required.

Answer C: Critical section.

OS10 defines a critical section as code accessing shared state that needs protection. The stem describes execution that may change shared data and therefore requires mutual exclusion, directly matching that definition. Deadlock and blocking are waiting states or failures, while termination means execution has ended. None identifies the protected shared-state code currently being executed.

Why each choice is right or wrong

A - Incorrect: Deadlock involves blocked dependencies, whereas the stem describes protected code actively accessing shared data.

B - Incorrect: A terminated process has ended execution, contradicting the stem’s currently executing shared-data operation.

C - Correct: Code that accesses shared data and requires mutual exclusion is a critical section.

D - Incorrect: A blocked process cannot proceed pending an event; the stem instead identifies the code requiring protection.

Primary note proof

Study-note section OS10

A race occurs when results depend on uncoordinated interleaving of conflicting accesses. A critical section is code accessing shared state that needs protection, not the resource itself. A correct exclusion solution requires mutual exclusion, progress and bounded waiting under stated assumptions.

Study-note section OS06

New processes are created/admitted; ready means eligible for CPU but not currently running. Running uses a CPU. Waiting/blocked means unable to proceed until an event; terminated means execution has ended.

Study-note section OS22

A set is deadlocked when each member waits for an event/resource that only another blocked member can provide.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Operating System module | PDF page 10 | lines 23-28

• Loads a program into memory.
• Executes the program.
• Handles program's execution.
• Provides a mechanism for process synchronization.
• Provides a mechanism for process communication.
• Provides a mechanism for deadlock handling.

Operating System module | PDF page 16 | lines 29-39

1. New The process is being created.
2.
Ready The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run.
3.
Running Process instructions are being executed (i.e. The process that is currently being executed).
4.
Waiting The process is waiting for some event to occur (such as the completion of an I/O operation).
5. Terminated The process has finished execution.

Question sources

Computer Science Exit Exam practice collection · Question 221

Question 179 | NS0308

Which one of the following function of operating system is categorized under process management?

A. Allocates the device in the efficient way

B. De-allocates memory when a process releases it.

C. Allocates the memory when the process requests it to do so.

D. Allocates the processor (CPU) to a process

Hint

Decide whether the process is eligible, running or waiting, then identify the responsible queue or scheduling role.

Answer D: Allocate the CPU to a process.

Allocating the CPU to a ready process is the scheduling role within process management. Device allocation belongs to device management. Allocating requested memory and reclaiming memory that has been released belong to memory management. Specifying the resource in the deallocation option removes its former overlap with several managers and leaves processor allocation as the distinct process-management function.

Why each choice is right or wrong

A - Incorrect: Allocating a device is a device-management responsibility.

B - Incorrect: Reclaiming released memory is a memory-management responsibility under the stated resource classification.

C - Incorrect: Allocating requested memory belongs to memory management.

D - Correct: Selecting a ready process to receive the CPU is the process-scheduling responsibility.

Primary note proof

Study-note section OS06

> Long-term scheduling admits jobs; medium-term scheduling suspends/swaps them; short-term scheduling selects ready work. The dispatcher performs the handoff/context restore after selection.

Study-note section OS-S03

> Memory management tracks which primary-memory regions are free or in use, assigns memory to processes and data, and deallocates it when no longer needed or when a process terminates. Allocation and deallocation concern memory ownership; CPU selection belongs to scheduling.

Study-note section OS01

> An OS allocates CPU time, memory, storage, devices and communication resources, and provides services to programs.

Question sources

Computer Science Exit Exam · 2018 · Question 29

Question 180 | NS0316

What is the primary function of an operating system?

A. To connect to the internet

B. To create documents and spreadsheets

C. To design computer hardware

D. To act as an intermediary between users and computer hardware

Hint

Separate broad resource-management goals from individual services or applications.

Answer D: Act as the service/resource intermediary between users/programs and hardware.

OS01 describes the OS allocating hardware resources and providing program services, with convenient interfaces and abstractions. D captures that intermediary role. Internet connectivity and document creation are narrower tasks, and hardware design is not a runtime OS service. The answer combines convenience and resource control rather than reducing the OS to one application or device function.

Why each choice is right or wrong

A - Incorrect: Internet access is one supported communication activity rather than the general purpose of an operating system.

B - Incorrect: Document and spreadsheet creation are application tasks that use OS services rather than define the OS.

C - Incorrect: Designing hardware is not the runtime resource/service function described for an operating system.

D - Correct: The OS provides interfaces and controlled resource services linking users/programs to the underlying hardware.

Primary note proof

Study-note section OS01

> An OS allocates CPU time, memory, storage, devices and communication resources, and provides services to programs. Convenience means making tasks easy through consistent interfaces, files and device abstractions. Efficiency means using resources effectively; it is not synonymous with the fastest response for every process. Application development tools use OS services but application development itself is not a core resource-management function.

Question sources

Computer Science Exit Exam · 2018 · Question 41

Question 181 | NS0323

A user installs a new printer. The OS automatically detects and configures it. This is an example of:

A. Plug-and-play functionality

B. Manual driver installation

C. Manual hardware management

D. Resource allocation

Hint

Separate device hardware, device-specific software and general event or transfer mechanisms.

Answer A: Plug-and-play functionality.

OS34 defines plug-and-play as discovering/configuring compatible new hardware, which directly matches automatic printer setup. A driver may still be needed, so automatic configuration does not imply driver-free operation. Manual driver installation and hardware management contradict the scenario's automatic action. Resource allocation may occur during setup but is broader than the specific feature named.

Why each choice is right or wrong

A - Correct: Automatic detection and configuration of the printer is the note's plug-and-play behavior.

B - Incorrect: The stem specifies automatic setup, not user-performed manual installation of a driver.

C - Incorrect: Manual hardware management does not match the automatic discovery/configuration described in the scenario.

D - Incorrect: Allocation can be part of setup, but it does not specifically name automatic device discovery/configuration.

Primary note proof

Study-note section OS34

> Plug-and-play discovers/configures compatible new hardware but may still require a driver.

Question sources

Computer Science Exit Exam · 2018 · Question 48

Question 182 | NS0330

In a system, resources cannot be forcibly taken from processes; they must be released voluntarily. Which deadlock condition does this represent?

A. No Preemption

B. Hold and Wait

C. Mutual Exclusion

D. Circular Wait

Hint

Trace the resource dependencies and distinguish necessary conditions from evidence of an actual deadlock.

Answer A: No preemption.

OS22 defines no resource preemption as voluntary release by the current holder. The stem repeats that resources cannot be forcibly taken, giving a direct match. Holding while requesting another resource, one-at-a-time ownership and a closed chain of waits are separate conditions and do not by themselves express the withdrawal restriction.

Why each choice is right or wrong

A - Correct: The holder must release voluntarily, so the system cannot forcibly preempt the resource.

B - Incorrect: Hold-and-wait concerns retaining one resource while requesting another rather than whether withdrawal is allowed.

C - Incorrect: Mutual exclusion limits simultaneous ownership, not whether allocated resources can be forcibly reclaimed.

D - Incorrect: Circular wait concerns a closed dependency chain, not the resource-release rule specified here.

Primary note proof

Study-note section OS22

> Mutual exclusion allows at most one holder of a nonshareable resource. Hold-and-wait means retaining a resource while requesting another. No resource preemption means release must be voluntary. Circular wait forms a closed chain of such waits.

Question sources

Computer Science Exit Exam · 2018 · Question 57

Question 183 | NS0359

How does an operating system improve efficiency in a multi-user environment?

A. By allowing only one user to access the system at a time

B. By disabling all background services

C. By using time-sharing and process scheduling to allocate CPU time fairly among users and processes

D. By requiring manual intervention for each task

Hint

Compare interaction, CPU sharing and deadline requirements under the stated system model.

Answer C: Time-sharing and scheduling distribute CPU service among users/processes.

OS02 describes time-sharing as rapid scheduling/preemption supporting interactive users. OS20 explains that fair allocation needs a share policy and may operate per user/group. C describes those relevant mechanisms rather than disabling services or requiring manual intervention. Its fairly should be understood under a specified policy, not as an automatic equality guarantee from time-sharing alone.

Why each choice is right or wrong

A - Incorrect: Restricting access to one user contradicts the multiuser setting rather than explaining its efficient sharing.

B - Incorrect: Disabling every background service is neither required nor the time-sharing/resource-allocation mechanism described.

C - Correct: Scheduling/time-sharing interleave useful work and can allocate service according to the system's fairness policy.

D - Incorrect: Manual intervention for each task adds administration rather than explaining efficient automatic CPU allocation.

Primary note proof

Study-note section OS02

> Multiprogramming keeps several jobs available so another can use the CPU while one waits; it does not require fixed time slices. Time-sharing uses rapid scheduling/preemption to support interactive users. A single CPU interleaves execution; multiple CPUs can execute work truly in parallel.

Study-note section OS20

> Fair-share scheduling allocates according to stated weights, potentially per user/group rather than per process. Equal per-process RR can let a user obtain more aggregate CPU by launching more processes. Proportional shares address that policy choice; fair need not mean equal.

Question sources

Computer Science Exit Exam · 2018 · Question 97

Question 184 | NS0684

What is the primary role of an Operating System?

A. Run application software only

B. Manage hardware and software resources and act as interface between user and hardware

C. Provide internet connectivity

D. Store user files permanently

Hint

Separate broad resource-management goals from individual services or applications.

Answer B: Manage resources and provide interfaces/services.

OS01 defines both allocation of CPU/memory/storage/devices/communications and services to programs, while describing convenient interfaces. B combines those core responsibilities. Running applications only, providing connectivity only or permanently storing files each captures a narrower task and omits the overall system-management role. The primary role is therefore the resource/interface combination, not a single service.

Why each choice is right or wrong

A - Incorrect: Application execution is supported, but only excludes the OS's other essential resource and service duties.

B - Correct: Resource management plus user/program interfaces matches the combined OS role defined here.

C - Incorrect: Internet connectivity is one communication service rather than the complete primary OS role.

D - Incorrect: Permanent file storage is a storage-device/filesystem service and not the complete operating-system purpose.

Primary note proof

Study-note section OS01

> An OS allocates CPU time, memory, storage, devices and communication resources, and provides services to programs. Convenience means making tasks easy through consistent interfaces, files and device abstractions. Efficiency means using resources effectively; it is not synonymous with the fastest response for every process. Application development tools use OS services but application development itself is not a core resource-management function.

Question sources

Operating Systems review questions · Question 1

Question 185 | NS0685

Which type of OS is designed to process large numbers of similar jobs without user interaction during execution?

A. Real-Time OS

B. Time-Sharing OS

C. Batch System

D. Single-User OS

Hint

Compare interaction, CPU sharing and deadline requirements under the stated system model.

Answer C: Batch system.

OS02 defines batch jobs as running with little interaction after submission. That matches processing submitted similar jobs without user interaction during execution. Time-sharing targets interactive users, real-time targets timing correctness, and single-user classifies concurrent access rather than job interaction. The job-processing clue therefore selects batch rather than a user-count or deadline category.

Why each choice is right or wrong

A - Incorrect: Real-time classification concerns meeting deadlines rather than the absence of user interaction during submitted job processing.

B - Incorrect: Time-sharing supports interactive users through rapid scheduling, unlike the noninteractive batch description.

C - Correct: Batch processing runs submitted jobs with little interaction, directly matching the scenario.

D - Incorrect: Single-user describes concurrent user access, not whether a workload is submitted and processed as batches.

Primary note proof

Study-note section OS02

> Batch jobs run with little interaction after submission. Multiprogramming keeps several jobs available so another can use the CPU while one waits; it does not require fixed time slices. Time-sharing uses rapid scheduling/preemption to support interactive users.

Study-note section OS02

> Single-user and multiuser describe concurrent user access, not the number of applications. Network systems expose remote services; distributed systems coordinate work across machines, sometimes hiding location. Real-time correctness includes timing: hard deadlines must be met, while soft deadline misses degrade service.

Question sources

Operating Systems review questions · Question 2

Question 186 | NS0686

In a Process Control Block (PCB), what does the Program Counter store?

A. The process priority level

B. The address of the next instruction to be executed for that process

C. The amount of CPU time used

D. The list of open files

Hint

Distinguish stored instructions, an executing instance and its saved execution state.

Answer B: The saved next-instruction position for resuming the process.

OS05 says the PCB records PC/registers alongside scheduling, memory/accounting and I/O information, and a saved PC identifies where execution resumes. Thus B describes its specific field role. Priority, CPU usage and open files can be other PCB information, but they are not the program-counter value. The question requires distinguishing fields within one record rather than excluding them from the PCB entirely.

Why each choice is right or wrong

A - Incorrect: Priority is scheduling information in the PCB, not the saved next-instruction position held by PC.

B - Correct: The saved PC records the instruction position at which the process will resume execution.

C - Incorrect: CPU time consumed is accounting information rather than the program-counter address.

D - Incorrect: Open files belong to I/O/open-resource state, not to the saved instruction-pointer field.

Primary note proof

Study-note section OS05

> A saved PC identifies where execution resumes.

Question sources

Operating Systems review questions · Question 3

Question 187 | NS0687

Which process state indicates that a process is waiting for an I/O event to complete?

A. Ready

B. Running

C. Waiting

D. Terminated

Hint

Decide whether the process is eligible, running or waiting, then identify the responsible queue or scheduling role.

Answer C: Waiting/blocked.

OS06 defines waiting/blocked as unable to proceed until an event and maps blocking I/O from running to waiting. Therefore a process waiting for I/O completion is in Waiting. It is not ready while the event prevents progress, not running during the wait, and not terminated merely because its request is outstanding. Completion normally makes it ready afterward.

Why each choice is right or wrong

A - Incorrect: Ready means eligible for CPU immediately; the outstanding blocking I/O prevents that eligibility.

B - Incorrect: Running means currently using a CPU, unlike the blocked interval awaiting I/O completion.

C - Correct: Waiting/blocked is the state for a process that cannot proceed until its I/O event completes.

D - Incorrect: Terminated means execution has ended, not temporarily paused pending completion of an I/O request.

Primary note proof

Study-note section OS06

> New processes are created/admitted; ready means eligible for CPU but not currently running. Running uses a CPU. Waiting/blocked means unable to proceed until an event; terminated means execution has ended. Running→ready follows time-slice expiry or another preemption. Running→waiting follows blocking I/O. I/O completion normally gives waiting→ready, not a guaranteed immediate running state.

Question sources

Operating Systems review questions · Question 4

Question 188 | NS0688

What is the primary purpose of a Context Switch?

A. To permanently stop a process

B. To save the state of the current process and restore the state of the next scheduled process

C. To allocate more memory to a process

D. To terminate a low-priority process

Hint

Distinguish stored instructions, an executing instance and its saved execution state.

Answer B: Save one execution context and restore another.

OS05 directly defines a context switch as saving one execution context and restoring another, with overhead interrupting useful application work. B matches. A process may later resume, so switching does not permanently stop or kill it. Allocating memory is another management operation, and a privilege transition alone need not switch processes.

Why each choice is right or wrong

A - Incorrect: The outgoing process can resume later; permanent stopping is termination rather than context switching.

B - Correct: Saving current state and restoring the selected execution context is the defined context-switch purpose.

C - Incorrect: Memory allocation changes resource assignment, not the saved/restored execution context defining a switch.

D - Incorrect: A context switch need not terminate any process, regardless of priority.

Primary note proof

Study-note section OS05

> A context switch saves one execution context and restores another; useful application work is interrupted by this overhead. An interrupt or system call can switch privilege without switching process.

Question sources

Operating Systems review questions · Question 5

Question 189 | NS0689

Which scheduler is responsible for selecting which process from the ready queue should be executed next on the CPU?

A. Long-Term Scheduler

B. Medium-Term Scheduler

C. Short-Term (CPU) Scheduler

D. I/O Scheduler

Hint

Decide whether the process is eligible, running or waiting, then identify the responsible queue or scheduling role.

Answer C: Short-term (CPU) scheduler.

OS06 assigns admission to the long-term scheduler, suspension/swapping to the medium-term scheduler and ready-work selection to the short-term scheduler. The question specifically asks which ready process receives the CPU next, so C is exact. The dispatcher then performs the handoff; that is distinct from selecting the ready process. Device request ordering is not CPU ready-queue selection.

Why each choice is right or wrong

A - Incorrect: The long-term scheduler admits jobs rather than making each immediate ready-queue CPU selection.

B - Incorrect: The medium-term scheduler suspends/swaps work rather than choosing the next CPU-ready process.

C - Correct: The short-term CPU scheduler selects the next eligible process from ready work.

D - Incorrect: An I/O scheduler orders device work rather than selecting the next process for the CPU.

Primary note proof

Study-note section OS06

> The ready queue contains eligible work; device queues hold blocked requests. The broad textbook job queue contains processes in the system, while an admission queue more narrowly holds jobs awaiting memory/admission. Long-term scheduling admits jobs; medium-term scheduling suspends/swaps them; short-term scheduling selects ready work. The dispatcher performs the handoff/context restore after selection.

Question sources

Operating Systems review questions · Question 6

Question 190 | NS0690

What is the role of the Medium-Term Scheduler?

A. Decide which jobs enter the system

B. Pick the next process for the CPU

C. Swap processes between main memory and secondary storage to manage memory

D. Handle device driver requests

Hint

Decide whether the process is eligible, running or waiting, then identify the responsible queue or scheduling role.

Answer C: Suspend/swap processes between resident and backing-store states.

OS06 explicitly assigns suspension/swapping to the medium-term scheduler. OS27 distinguishes traditional whole-process swapping to backing storage from page movement. Thus C is the matching memory/load-control role. Admitting jobs belongs to long-term scheduling, picking the next ready process to short-term scheduling, and servicing drivers to device-management paths.

Why each choice is right or wrong

A - Incorrect: Deciding admission into the system is the long-term scheduler's role in the supplied model.

B - Incorrect: Selecting the next CPU process belongs to the short-term scheduler, not the medium-term one.

C - Correct: Medium-term scheduling suspends/swaps processes to control resident load and memory use.

D - Incorrect: Device-driver requests concern I/O service rather than the scheduler's suspension/swapping function.

Primary note proof

Study-note section OS06

> Long-term scheduling admits jobs; medium-term scheduling suspends/swaps them; short-term scheduling selects ready work.

Study-note section OS27

> Swapping traditionally moves a whole process to backing store; paging moves individual pages.

Study-note section OS34

> A driver converts OS requests into device-specific operations.

Question sources

Operating Systems review questions · Question 7

Question 191 | NS0691

A Race Condition occurs when:

A. Two processes finish execution at the same time

B. Multiple processes access shared resources simultaneously without proper synchronization, causing unpredictable results

C. A high-priority process always runs before a low-priority one

D. A process is preempted by the OS scheduler

Hint

Identify conflicting shared accesses and the safety or progress property required.

Answer B: Uncoordinated conflicting shared accesses make results depend on interleaving.

OS10 defines a race by results depending on uncoordinated interleaving of conflicting accesses. B is the intended match, with shared access understood to include a conflict such as an update; simultaneous read-only access alone is insufficient. Coincident completion, ordinary priority selection and preemption do not by themselves establish a shared-state race.

Why each choice is right or wrong

A - Incorrect: Two completions at the same time need not involve conflicting shared accesses or interleaving-dependent results.

B - Correct: Unsynchronized conflicting shared accesses can make results depend on execution interleaving, which is the race condition.

C - Incorrect: Priority order alone is a scheduling choice and does not establish a conflicting shared-data race.

D - Incorrect: Preemption can expose a race but is not itself sufficient without conflicting unprotected accesses.

Primary note proof

Study-note section OS10

> A race occurs when results depend on uncoordinated interleaving of conflicting accesses.

Study-note section OS10

> Two threads read counter=0, each computes 1 and each stores 1. Two increments produce 1 instead of 2. Protecting the entire read-modify-write with one lock prevents that lost update.

Question sources

Operating Systems review questions · Question 8

Question 192 | NS0692

Which of the following is NOT a necessary condition for deadlock?

A. Mutual Exclusion

B. Hold and Wait

C. Preemption

D. Circular Wait

Hint

Trace the resource dependencies and distinguish necessary conditions from evidence of an actual deadlock.

Answer C: Preemption is not a necessary condition; no-preemption is.

OS22 names mutual exclusion, hold-and-wait, no resource preemption and circular wait as the necessary set. The option Preemption reverses one of those conditions. Therefore C is the exception, while the other three are genuine members. This resource-preemption rule is separate from whether CPU scheduling uses timer preemption.

Why each choice is right or wrong

A - Incorrect: Mutual exclusion is explicitly one of the four necessary conditions in the standard resource model.

B - Incorrect: Hold-and-wait is explicitly necessary: a process retains a resource while requesting another.

C - Correct: The required condition is no-preemption; permitting preemption is not one of the necessary conditions.

D - Incorrect: Circular wait is explicitly a required closed dependency chain in the standard model.

Primary note proof

Study-note section OS22

> Mutual exclusion allows at most one holder of a nonshareable resource. Hold-and-wait means retaining a resource while requesting another. No resource preemption means release must be voluntary. Circular wait forms a closed chain of such waits.

Study-note section OS22

> All four are necessary in the standard reusable-resource model.

Question sources

Operating Systems review questions · Question 9

Question 193 | NS0693

All three processes arrive at time 0 and are ready immediately. Their CPU burst times are P1 = 24 ms, P2 = 3 ms and P3 = 3 ms. Using non-preemptive FCFS with tie order P1, P2, P3, and ignoring context-switch overhead, what is the average waiting time?

A. 3 ms

B. 9 ms

C. 17 ms

D. 27 ms

Hint

Follow arrival order, include idle periods and distinguish waiting from completion time.

Answer C: 17 ms.

FCFS runs P1 from 0 to 24 ms, P2 from 24 to 27 ms, and P3 from 27 to 30 ms. Because every process arrives at 0, their waiting times are 0, 24 and 27 ms. The average is (0+24+27)/3 = 51/3 = 17 ms, so C is correct. The average turnaround is a different quantity: (24+27+30)/3 = 27 ms.

Why each choice is right or wrong

A - Incorrect: 3 ms is the burst time of P2 or P3, not the average ready-queue wait.

B - Incorrect: The three waits total 51 ms, so dividing by three cannot give 9 ms.

C - Correct: The waits 0, 24 and 27 ms average to 17 ms.

D - Incorrect: 27 ms is the average turnaround time in this example, not the average waiting time.

Primary note proof

Study-note section OS15

> FCFS runs ready bursts in arrival order and ordinarily does not preempt.

Study-note section OS15

> All ready at time 0, tie order P1=24,P2=3,P3=3: schedule 0-24 P1,24-27 P2,27-30 P3. Waiting times 0,24,27 average to 17. Turnarounds 24,27,30 average to 27. If arrivals are not simultaneous, recompute using their actual times.

Study-note section OS13

> For arrival A, first start S and completion C, response=S−A and turnaround=C−A. For a single CPU burst B with no I/O, waiting=C−A−B.

Question sources

Operating Systems review questions · Question 10

Question 194 | NS0694

What problem does the 'Aging' technique solve in Priority Scheduling?

A. Deadlock

B. Starvation of low-priority processes

C. Race conditions

D. Memory fragmentation

Hint

Check priority direction, preemption and whether waiting can change effective rank.

Answer B: Starvation of low-priority processes.

OS17 defines aging as increasing waiting jobs' effective priority to reduce indefinite postponement under suitable rules/workloads. Thus B names the problem it addresses. It does not remove a deadlock dependency cycle, coordinate conflicting memory updates or consolidate fragmented memory. The result still depends on the priority/aging policy; naming aging alone does not provide an unconditional deadline guarantee.

Why each choice is right or wrong

A - Incorrect: Aging changes priority rather than breaking the blocked resource dependencies of a deadlock.

B - Correct: Increasing effective priority of waiting low-priority jobs addresses their starvation under sustained higher-priority load.

C - Incorrect: Races require shared-access synchronization; priority aging is not a mutual-exclusion mechanism.

D - Incorrect: Memory fragmentation concerns allocation layout rather than the priority of waiting CPU jobs.

Primary note proof

Study-note section OS17

> Aging increases waiting jobs' effective priority according to a rule, reducing indefinite postponement when the priority scheme and workload permit. It is not a deadlock cure.

Study-note section OS10

> A race occurs when results depend on uncoordinated interleaving of conflicting accesses.

Study-note section OS29

> Internal fragmentation is unused space inside an allocated unit. External fragmentation is sufficient total free space split into holes that cannot satisfy a required contiguous allocation.

Question sources

Operating Systems review questions · Question 11

Question 195 | NS0695

Which scheduling algorithm is known as preemptive SJF?

A. Round Robin

B. FCFS

C. Shortest Remaining Time First (SRTF)

D. Priority Scheduling

Hint

Compare the ready jobs using the stated burst or remaining-time rule.

Answer C: Shortest Remaining Time First.

OS16 distinguishes non-preemptive SJF/SJN from SRTF, which reconsiders at arrivals and can preempt for shorter remaining work. Therefore SRTF is the preemptive SJF variant. RR uses fixed cyclic turns, FCFS arrival order and generic priority an independently defined ranking. The original burst length and the remaining length must not be confused after a job has executed.

Why each choice is right or wrong

A - Incorrect: RR selects cyclic quantum turns rather than the shortest remaining CPU work.

B - Incorrect: FCFS follows arrival order and is ordinarily non-preemptive rather than preemptive shortest-job scheduling.

C - Correct: SRTF chooses minimum remaining work and can interrupt a running job for a shorter arrival.

D - Incorrect: Generic priority scheduling does not specifically rank by remaining burst length and can be either variant.

Primary note proof

Study-note section OS16

> Non-preemptive SJF/SJN chooses the shortest next burst among currently ready jobs. SRTF reconsiders on arrivals and selects the shortest remaining work; a shorter newcomer can preempt. Future burst lengths normally need estimates.

Study-note section OS18

> RR serves a ready queue cyclically for at most quantum q per turn.

Study-note section OS15

> FCFS runs ready bursts in arrival order and ordinarily does not preempt.

Question sources

Operating Systems review questions · Question 12

Question 196 | NS0696

In Round Robin scheduling, what is the key challenge?

A. Managing priority levels

B. Determining the appropriate time quantum length

C. Avoiding deadlocks

D. Allocating memory to processes

Hint

Trace bounded turns in ready-queue order and include the stated overhead and arrivals.

Answer B: Choose an appropriate time quantum.

OS18 says a large RR quantum approaches FCFS, while tiny quanta increase context-switch overhead. Selecting q therefore trades responsiveness against overhead, matching B. Priority management, deadlock policy and memory allocation are separate concerns rather than the defining parameter-selection challenge of ordinary Round Robin. The useful quantum depends on workloads and switching cost.

Why each choice is right or wrong

A - Incorrect: Basic RR uses cyclic ready-queue turns rather than requiring a hierarchy of priority levels.

B - Correct: Quantum length controls responsiveness and switching overhead, creating RR's characteristic tuning tradeoff.

C - Incorrect: Deadlock avoidance concerns resources and dependencies, not the defining RR quantum parameter.

D - Incorrect: Memory allocation is a separate management problem rather than the Round Robin selection rule's main parameter.

Primary note proof

Study-note section OS18

> A large quantum approaches FCFS; tiny quanta increase switch overhead.

Study-note section OS18

> With n continuously ready jobs, no priority bypass and zero overhead, a just-requeued job waits at most (n−1)q for its next turn. This bounds one interval, not total lifetime waiting when arrivals or blocking are unrestricted. Include context-switch costs if given.

Question sources

Operating Systems review questions · Question 13

Question 197 | NS0697

What is a Semaphore?

A. A scheduling algorithm

B. A memory management technique

C. A synchronization variable that controls access to shared resources in a multi-process environment

D. A type of file allocation method

Hint

Track available permits and blocked operations using the stated semaphore model.

Answer C: A synchronization variable with atomic wait and signal operations.

OS11 defines a semaphore through atomic wait/P/down and signal/V/up operations. Tokens control whether participants can proceed with shared resource work, with binary or counting forms. C identifies that synchronization role. Scheduling, memory-management and file-allocation policies solve different resource-selection/layout problems and do not define semaphore semantics.

Why each choice is right or wrong

A - Incorrect: A semaphore synchronizes participants; it is not itself a CPU-selection scheduling algorithm.

B - Incorrect: It controls coordination through permits, rather than choosing memory locations or translation layouts.

C - Correct: Atomic wait/signal on a synchronization variable controls permits for concurrent shared-resource operations.

D - Incorrect: File allocation chooses storage-block layout; that is separate from semaphore synchronization semantics.

Primary note proof

Study-note section OS11

> A semaphore provides atomic wait/P/down and signal/V/up operations. Under the nonnegative-token model, wait consumes one available token or blocks when none is available; signal supplies a token and may wake a waiter. Implementations using negative internal counts encode waiters differently, so state the model. A binary semaphore represents one permit; a counting semaphore tracks several. A mutex normally has ownership rules that a general semaphore lacks.

Study-note section OS15

> FCFS runs ready bursts in arrival order and ordinarily does not preempt.

Study-note section OS28

> A contiguous allocation gives a process one continuous region.

Study-note section OS38

> Contiguous allocation uses consecutive blocks; logical block i is start+i and supports direct access, but growth and external fragmentation are problems.

Question sources

Operating Systems review questions · Question 14

Question 198 | NS0698

What is the difference between a Binary Semaphore and a Counting Semaphore?

A. Binary semaphores are faster

B. Binary semaphores have only two states (0 or 1), while counting semaphores track available resources up to a maximum count

C. Counting semaphores are used only for deadlock prevention

D. There is no difference

Hint

Track available permits and blocked operations using the stated semaphore model.

Answer B: Binary tracks one permit; counting tracks several.

OS11 defines a binary semaphore as one permit and a counting semaphore as several, under the nonnegative-token model. Thus binary availability is 0/1 while a resource pool can have a larger count. B states this logical distinction. Internal negative-waiter encodings are implementation details, not a reason to claim speed guarantees, deadlock-only usage or no difference.

Why each choice is right or wrong

A - Incorrect: The note gives no universal relative-speed guarantee; permit capacity is the defining distinction.

B - Correct: In the nonnegative-token model, binary availability is zero/one while counting semaphores track several available permits.

C - Incorrect: Counting semaphores coordinate resource pools and are not limited to preventing deadlock.

D - Incorrect: The allowed permit count is a substantive difference explicitly stated here.

Primary note proof

Study-note section OS11

> A semaphore provides atomic wait/P/down and signal/V/up operations. Under the nonnegative-token model, wait consumes one available token or blocks when none is available; signal supplies a token and may wake a waiter. Implementations using negative internal counts encode waiters differently, so state the model. A binary semaphore represents one permit; a counting semaphore tracks several. A mutex normally has ownership rules that a general semaphore lacks.

Question sources

Operating Systems review questions · Question 15

Question 199 | NS0699

What must hold simultaneously for a deadlock to occur?

A. Only Mutual Exclusion and Circular Wait

B. Any two of the four conditions

C. All four necessary conditions: Mutual Exclusion, Hold-and-Wait, No Preemption, and Circular Wait

D. Only Circular Wait

Hint

Trace the resource dependencies and distinguish necessary conditions from evidence of an actual deadlock.

Answer C: All four standard necessary conditions.

OS22 says all four conditions are necessary in the standard reusable-resource model. C names that full set. Any two or a cycle alone does not state the complete conjunction. This is a necessary-condition answer, not an assertion that merely allowing all four policies proves a deadlock already exists in a particular allocation state.

Why each choice is right or wrong

A - Incorrect: It omits hold-and-wait and no-preemption, both necessary in the stated standard resource model.

B - Incorrect: An arbitrary pair does not supply all four necessary restrictions for the standard deadlock model.

C - Correct: This option lists mutual exclusion, hold-and-wait, no-preemption and circular wait together.

D - Incorrect: A cycle alone omits other necessary restrictions; it is not the complete standard condition set.

Primary note proof

Study-note section OS22

> Mutual exclusion allows at most one holder of a nonshareable resource. Hold-and-wait means retaining a resource while requesting another. No resource preemption means release must be voluntary. Circular wait forms a closed chain of such waits.

Study-note section OS22

> All four are necessary in the standard reusable-resource model. Their availability as general system policies does not prove a deadlock exists at this instant; inspect actual allocations/requests.

Question sources

Operating Systems review questions · Question 16

Question 200 | NS0700

In a Resource Allocation Graph (RAG), what does a cycle with single-instance resources indicate?

A. Potential deadlock only

B. No deadlock possible

C. Definite deadlock

D. High CPU utilization

Hint

Follow request and assignment arrows and check how many instances each resource has.

Answer C: Definite deadlock under single-instance resource assumptions.

OS23 says a cycle implies deadlock when every resource type has one instance. The stem supplies that decisive assumption, so C follows. With multiple instances a cycle would indicate only a possibility, but importing that different model would ignore the literal question. A graph cycle also says nothing by itself about high CPU utilization.

Why each choice is right or wrong

A - Incorrect: Potential-only applies to the more general multiple-instance caveat, not the supplied single-instance case.

B - Incorrect: A cycle in the stated single-instance model establishes deadlock rather than excluding it.

C - Correct: The note explicitly gives cycle implies deadlock for a graph with one instance per resource type.

D - Incorrect: Resource dependency cycles do not measure CPU busy time or establish high processor utilization.

Primary note proof

Study-note section OS23

> In a resource-allocation graph, P→R is a request and an instance R→P is an assignment. Follow these directions to form waits. No cycle implies no deadlock; with one instance of every resource type, a cycle implies deadlock. With multiple instances a cycle alone is insufficient: another instance may permit completion. A wait-for graph replaces single-instance resource nodes with P→Q when P waits for Q.

Question sources

Operating Systems review questions · Question 17

Question 201 | NS0701

Which deadlock handling method allows deadlock to occur but then recovers from it?

A. Deadlock Prevention

B. Deadlock Ignorance

C. Deadlock Recovery

D. Deadlock Avoidance

Hint

Separate handling before allocation from actions taken after detection, and consider the cost of checking.

Answer C: Deadlock recovery.

OS26 describes recovery through terminating selected processes or safely reclaiming resources and rolling work back after deadlock is detected. That matches allowing occurrence and then resolving it. Prevention removes a necessary condition, avoidance preserves safety before granting, and ignoring accepts risk without committing to recovery. The complete handling strategy is detection plus recovery; C is the offered recovery component.

Why each choice is right or wrong

A - Incorrect: Prevention constrains requests so the targeted deadlock class cannot arise rather than accepting occurrence first.

B - Incorrect: Ignoring rare deadlock does not define active detection and subsequent recovery.

C - Correct: Recovery acts on an actual detected deadlock using termination, reclamation or rollback as appropriate.

D - Incorrect: Avoidance checks safe states before grants rather than deliberately permitting deadlock and then repairing it.

Primary note proof

Study-note section OS26

> Detection asks whether the current state is deadlocked. Recovery can terminate selected processes or reclaim safely preemptable resources and roll affected work back to a checkpoint. Termination and rollback are both possible recovery tools; choose according to the resource and process conditions.

Study-note section OS24

> Prevention constrains how requests are made.

Study-note section OS25

> Avoidance allows requests only if the resulting state remains safe.

Study-note section OS26

Ignoring a sufficiently rare problem is sometimes called the ostrich policy, but real systems still prevent or detect selected deadlocks.

Question sources

Operating Systems review questions · Question 18

Question 202 | NS0702

What is Address Binding at 'Load Time'?

A. Addresses are fixed by the compiler before execution

B. Addresses are determined while the program executes

C. Final memory addresses are assigned when the program is loaded into memory (since addresses are unknown at compile time)

D. Addresses are stored on the hard disk

Hint

Track when an address or library reference is resolved and the direction of translation.

Answer C: Relocate addresses when the program is loaded.

OS27 defines load-time binding as relocation when placed into memory. That matches C's final-address assignment at load when placement was unknown at compile time. Fixing absolute addresses during compilation is compile-time binding; translating while running is execution-time binding. Merely storing address bytes on a disk describes neither the placement decision nor its timing.

Why each choice is right or wrong

A - Incorrect: Fixing absolute addresses in the compiler is compile-time binding, not relocation on loading.

B - Incorrect: Translation during execution is execution-time binding rather than assignment during loading.

C - Correct: The loader resolves placement-dependent addresses when placing the program into memory.

D - Incorrect: Storing address information on disk does not define when placement-dependent address binding occurs.

Primary note proof

Study-note section OS27

> Compile-time binding uses absolute addresses if placement is known. Load-time binding relocates when placed into memory. Execution-time binding translates while running and permits movement with hardware support.

Question sources

Operating Systems review questions · Question 19

Question 203 | NS0703

Under basic page-frame allocation and variable-segment allocation, which statement correctly contrasts allocation-unit size and external fragmentation?

A. Paging uses variable-sized blocks; segmentation uses fixed-sized blocks

B. Paging divides memory into fixed-sized pages eliminating external fragmentation; segmentation uses variable-sized logical segments but can cause external fragmentation

C. Segmentation is used only in virtual memory

D. Paging requires no hardware support

Hint

Check the logical region and offset bound before computing its physical location.

Answer B: Fixed-size paging avoids external page-frame fragmentation; variable segments can fragment externally.

Paging maps fixed-size pages into equal-size frames, avoiding external fragmentation of frame allocation, although internal waste can remain. Segmentation uses variable-length logical regions and can leave external holes. This is the unit-size and fragmentation contrast in B. A reverses the sizes. C does not supply that contrast, and D substitutes a hardware-support claim for it; page translation still requires mappings and checks.

Why each choice is right or wrong

A - Incorrect: The sizes are reversed: pages are fixed-size units and segments are variable-length regions.

B - Correct: This gives the required fixed-page versus variable-segment size and external-fragmentation contrast.

C - Incorrect: An exclusive-use statement does not compare allocation-unit sizes or external fragmentation as requested.

D - Incorrect: This does not give the requested size/fragmentation contrast; paging also uses translation mappings and checks.

Primary note proof

Study-note section OS31

> Under the basic models, paging uses fixed units/page tables and can waste space internally; segmentation uses variable logical units/segment tables and can fragment externally. Segments reflect program structure whereas ordinary pages are largely transparent to applications.

Study-note section OS30

> Pages and physical frames have equal fixed size P. Divide virtual address v into page=floor(v/P) and offset=v mod P. The page table maps a page to a frame and protection/status flags. Physical address=frame×P+offset.

Question sources

Operating Systems review questions · Question 20

Question 204 | NS0704

What causes External Fragmentation in memory management?

A. Allocated memory blocks being larger than needed

B. Free memory being scattered in small non-contiguous pieces making it impossible to satisfy large allocation requests

C. Processes using too much CPU time

D. Too many open files in the directory

Hint

Locate unused space relative to an allocated unit and distinguish total free space from contiguous free space.

Answer B: Free capacity is split across holes too small for a required contiguous allocation.

OS29 defines external fragmentation as sufficient total free space split into holes unable to satisfy a contiguous allocation. B matches that definition, with large requests understood to require one continuous region. A instead describes internal fragmentation. CPU time and open-file counts may create other pressures but do not define the spatial fragmentation of free memory.

Why each choice is right or wrong

A - Incorrect: Unused space within an oversized allocated block is internal fragmentation, not external free-hole fragmentation.

B - Correct: Noncontiguous small holes can have enough total capacity yet fail to satisfy one contiguous request.

C - Incorrect: CPU execution time does not define the spatial layout of free memory holes.

D - Incorrect: Open-file count is a resource-use limit rather than the definition of external memory fragmentation.

Primary note proof

Study-note section OS29

> Internal fragmentation is unused space inside an allocated unit. External fragmentation is sufficient total free space split into holes that cannot satisfy a required contiguous allocation.

Question sources

Operating Systems review questions · Question 21

Question 205 | NS0705

What is the purpose of the Page Table in Paging?

A. To allocate CPU time to processes

B. To map logical page numbers to physical frame numbers in memory

C. To store file access permissions

D. To manage disk I/O operations

Hint

Split the address into page and offset, then apply the mapping and access checks.

Answer B: Map virtual/logical pages to physical frames.

OS30 says the page table maps each virtual page to a frame with protection/status flags. B directly states the primary translation role. CPU time allocation belongs to scheduling; file permissions belong to filesystem metadata, despite page tables also having memory-protection bits. Disk I/O may service nonresident pages but does not define the page table's mapping purpose.

Why each choice is right or wrong

A - Incorrect: CPU time allocation is scheduling, not the page-to-frame translation recorded by a page table.

B - Correct: The table identifies the physical frame corresponding to each virtual page and records relevant flags.

C - Incorrect: Memory access-protection flags differ from filesystem file-access permissions; file permissions are not the page table's purpose.

D - Incorrect: Page faults may cause I/O, but general disk-operation management is not the table's translation role.

Primary note proof

Study-note section OS30

> Pages and physical frames have equal fixed size P. Divide virtual address v into page=floor(v/P) and offset=v mod P. The page table maps a page to a frame and protection/status flags.

Study-note section OS36

> A file has content and metadata such as name, size, owner, permissions and timestamps. open obtains an open-file reference, read/write transfer bytes, seek changes an offset, close releases a reference, truncate changes size and unlink removes a name.

Question sources

Operating Systems review questions · Question 22

Question 206 | NS0706

In Segmentation, how is a physical address calculated?

A. Page number x frame size

B. Base address of segment + offset

C. Logical address - limit register

D. Frame number + page offset

Hint

Check the logical region and offset bound before computing its physical location.

Answer B: Physical address = segment base + valid offset.

OS31 gives segment-table base, limit and permissions, then validates (segment,offset) and adds base+offset. B matches, conditional on the offset passing the bound/protection checks. A and D mix in paging components and omit required scaling or displacement, while subtracting a limit is not the segment translation formula. An invalid offset traps rather than spilling into another segment.

Why each choice is right or wrong

A - Incorrect: Page number times frame size is not the segment base-plus-offset translation and omits required mapping.

B - Correct: After segment and bound checks, add the selected segment's physical base to the offset.

C - Incorrect: The limit is a bound for validation, not a number subtracted to form the physical address.

D - Incorrect: Frame terminology belongs to paging, and adding an unscaled frame number is not the segmentation formula.

Primary note proof

Study-note section OS31

> Segmentation uses logical variable-length units such as code, stack and arrays. A segment-table entry provides base, limit and permissions. For (segment,offset), validate the segment and check the bound, then add base+offset. An out-of-range offset traps; it must not spill into the next segment.

Study-note section OS30

> Pages and physical frames have equal fixed size P. Divide virtual address v into page=floor(v/P) and offset=v mod P. The page table maps a page to a frame and protection/status flags. Physical address=frame×P+offset.

Question sources

Operating Systems review questions · Question 23

Question 207 | NS0707

What is Virtual Memory?

A. Extra RAM chips added to a computer

B. A technique using hard disk space as an extension of physical memory, allowing programs larger than physical memory to execute

C. The memory inside the CPU cache

D. Memory used only by the operating system

Hint

Separate a permitted nonresident access from an illegal access and trace the required handling steps.

Answer B: Translated protected address spaces can use backing storage for nonresident portions.

OS32 describes virtual memory as protected translated address spaces with nonresident program portions backed by storage. B gives the disk-backed extension aspect and can allow programs exceeding resident physical memory to execute. It is not a complete definition of virtual memory, nor does it eliminate finite capacity constraints. The other alternatives confuse RAM installation, cache or an OS-only restriction.

Why each choice is right or wrong

A - Incorrect: Installing RAM chips adds physical capacity; virtual-memory translation is a different mechanism.

B - Correct: Backing storage holds nonresident portions, permitting a program's address space to exceed its currently resident physical memory.

C - Incorrect: CPU cache retains copied memory blocks and is not the virtual-address/backing-store mechanism.

D - Incorrect: Processes receive virtual address spaces; virtual memory is not restricted to storage used only by the OS.

Primary note proof

Study-note section OS32

> Virtual memory provides translated, protected address spaces and allows nonresident portions of a program to be backed by storage. It does not add physical RAM or remove all capacity constraints.

Study-note section OS35

> Caching retains copies to accelerate reuse.

Question sources

Operating Systems review questions · Question 24

Question 208 | NS0708

In demand paging, assume the accessed virtual page is valid, permitted and backed by disk, but its contents are not resident and must be read from disk. What event is the page fault in this scenario?

A. A hardware memory error

B. When the CPU tries to access a page not currently in physical memory, requiring the OS to load it from disk

C. An error in the page table

D. When a process exceeds its allocated memory

Hint

Separate a permitted nonresident access from an illegal access and trace the required handling steps.

Answer B: When the CPU tries to access a page not currently in physical memory, requiring the OS to load it from disk

A valid nonresident access traps so the OS can obtain the page contents, update the mapping and restart the access. The specified page is disk-backed and requires a read, making B accurate for this scenario. Page faults in general need not read disk; zero-fill and shared mappings are other cases. Nothing here requires hardware failure, table corruption or exceeding the permitted address range.

Why each choice is right or wrong

A - Incorrect: The trap does not imply damaged memory hardware.

B - Correct: The valid disk-backed page is nonresident, so this access requires the stated page-in.

C - Incorrect: A correct table may intentionally mark a valid page nonresident.

D - Incorrect: The page is explicitly within the valid permitted mapping, so no allocation violation is implied.

Primary note proof

Study-note section OS32

> Demand paging brings a page in when accessed. On a nonresident access: trap, validate mapping/permissions, find a frame or choose a victim, save a dirty victim if needed, obtain contents, update mapping and restart.

Study-note section OS32

> Not every fault reads a disk: some allocate zero-filled pages or establish shared mappings. An illegal access is rejected, not automatically loaded.

Question sources

Operating Systems review questions · Question 25

Question 209 | NS0709

What is Demand Paging?

A. Loading the entire program into memory before execution

B. Loading only the pages that are actually needed during program execution, keeping others on disk until required

C. Allocating memory in fixed partitions

D. Swapping entire processes between memory and disk

Hint

Separate a permitted nonresident access from an illegal access and trace the required handling steps.

Answer B: Bring a page into memory when it is needed/accessed.

OS32 defines demand paging as bringing a page in when accessed. B matches loading only the pages demanded during execution, with disk backing for the described stored portions. Not all pages must literally originate on disk because zero-fill/shared cases exist. Whole-program loading, fixed contiguous partitions and whole-process swapping do not name this per-page demand policy.

Why each choice is right or wrong

A - Incorrect: Loading the entire program before execution is eager loading rather than fetching pages on demand.

B - Correct: Pages are brought in as execution needs them; disk backing is the described common stored-page case.

C - Incorrect: Fixed partitions are a contiguous-allocation arrangement rather than the page-access-triggered loading policy.

D - Incorrect: Moving an entire process is traditional swapping, not the individual-page demand mechanism.

Primary note proof

Study-note section OS32

> Demand paging brings a page in when accessed. On a nonresident access: trap, validate mapping/permissions, find a frame or choose a victim, save a dirty victim if needed, obtain contents, update mapping and restart.

Study-note section OS27

> Swapping traditionally moves a whole process to backing store; paging moves individual pages.

Study-note section OS28

> A contiguous allocation gives a process one continuous region.

Question sources

Operating Systems review questions · Question 26

Question 210 | NS0710

What is the difference between Internal and External Fragmentation?

A. Both are the same type of memory waste

B. Internal fragmentation wastes space inside allocated partitions (allocated > needed); external fragmentation wastes space between allocations

C. External fragmentation occurs in paging systems

D. Internal fragmentation only occurs in virtual memory

Hint

Locate unused space relative to an allocated unit and distinguish total free space from contiguous free space.

Answer B: Internal waste is inside allocations; external holes are between them.

OS29 defines internal fragmentation as unused space inside an allocated unit and external fragmentation as free holes unable to serve a required contiguous request. B states the location distinction. C is not the general definition and needs allocator scope; D adds an unsupported virtual-memory-only restriction. These wastes can arise from allocation geometry rather than the program's CPU behavior.

Why each choice is right or wrong

A - Incorrect: The note explicitly distinguishes internal unused allocation space from external free-hole fragmentation.

B - Correct: Internal space is wasted within an allocation; external waste is free space split between allocations.

C - Incorrect: This does not define the contrast; basic page-frame allocation avoids external fragmentation, though other allocators may differ.

D - Incorrect: Unused space within an allocated unit is not limited by definition to virtual-memory systems.

Primary note proof

Study-note section OS29

> Internal fragmentation is unused space inside an allocated unit. External fragmentation is sufficient total free space split into holes that cannot satisfy a required contiguous allocation.

Study-note section OS29

> Basic fixed-frame paging avoids external fragmentation of page-frame allocation but can waste the final page internally.

Question sources

Operating Systems review questions · Question 27

Question 211 | NS0711

Which memory allocation strategy picks the first hole large enough to accommodate a process?

A. Best-Fit

B. Worst-Fit

C. First-Fit

D. Next-Fit

Hint

Apply the stated hole-selection or base-and-bound rule with its explicit convention.

Answer C: First-fit.

OS28 defines first-fit as selecting the first adequate hole. This is exactly the stem's first hole large enough. Best-fit searches for the smallest adequate hole, worst-fit for the largest, and next-fit resumes from a prior search position. The explicit first-hole condition selects first-fit without any numeric hole trace.

Why each choice is right or wrong

A - Incorrect: Best-fit minimizes adequate hole size rather than stopping at the first adequate hole.

B - Incorrect: Worst-fit selects the largest available hole rather than the first adequate one encountered.

C - Correct: First-fit stops at the first hole that can accommodate the requested allocation.

D - Incorrect: Next-fit continues from the previous search position; the stem gives the ordinary first-fit criterion.

Primary note proof

Study-note section OS28

> First-fit selects the first adequate hole; next-fit continues from the previous search position; best-fit selects the smallest adequate hole; worst-fit selects the largest.

Question sources

Operating Systems review questions · Question 28

Question 212 | NS0712

What is Swapping in OS memory management?

A. Exchanging CPU registers between processes

B. Temporarily moving an entire process from main memory to a backing store (disk) to free memory for other processes

C. Copying files between directories

D. Replacing old CPU instructions with new ones

Hint

Track when an address or library reference is resolved and the direction of translation.

Answer B: Temporarily move a whole process to backing store.

OS27 explicitly distinguishes traditional swapping of a whole process to backing store from paging individual pages and context switching execution state. B describes the whole-process memory-management operation. Saving registers is a context-switch task, file copying is a filesystem operation, and replacing instruction contents does not define swapping's residence change.

Why each choice is right or wrong

A - Incorrect: Saving/restoring CPU register contexts is context switching rather than moving a whole process out of RAM.

B - Correct: Traditional swapping moves a process image to backing storage to free resident memory.

C - Incorrect: Copying files changes directory/storage contents rather than a process image's memory residence.

D - Incorrect: Replacing instruction contents is not the defined transfer of a whole process between RAM and backing store.

Primary note proof

Study-note section OS27

> Swapping traditionally moves a whole process to backing store; paging moves individual pages. These differ from a context switch, which primarily changes execution state.

Study-note section OS05

> A context switch saves one execution context and restores another; useful application work is interrupted by this overhead.

Question sources

Operating Systems review questions · Question 29

Question 213 | NS0713

What is Dynamic Loading?

A. Loading the entire program into memory at startup

B. Loading routines from disk into memory only when they are actually called during execution

C. Dynamically changing the priority of a process

D. Moving pages between RAM and disk automatically

Hint

Track when an address or library reference is resolved and the direction of translation.

Answer B: Load a routine only when needed.

OS27 directly defines dynamic loading as bringing a routine in when needed. B states the routine-on-call form of that rule. Eager startup loading is the opposite, priority changes are scheduling, and automatic page movement is a virtual-memory mechanism at a different granularity. Dynamic loading concerns when code routines become resident, not which process gets CPU priority.

Why each choice is right or wrong

A - Incorrect: Loading all routines at startup is eager loading rather than bringing them in only on demand.

B - Correct: A routine is loaded when execution actually calls or needs it, matching dynamic loading.

C - Incorrect: Changing priority affects scheduling and does not load a routine's code into memory.

D - Incorrect: Paging moves pages by virtual-memory demand, whereas dynamic loading names demand-driven routine loading.

Primary note proof

Study-note section OS27

> Dynamic loading brings a routine in when needed.

Study-note section OS32

> Demand paging brings a page in when accessed.

Study-note section OS17

> Priority scheduling chooses the highest-priority ready job.

Question sources

Operating Systems review questions · Question 30

Question 214 | NS0714

What is the key management-layer distinction between user-level threads and kernel-level threads, without assuming universal speed or reliability advantages?

A. User-level threads are managed by a user-space library; kernel-level threads are known to and managed by the OS

B. Kernel-level threads are always faster

C. User-level threads can access hardware directly

D. There is no practical difference between them

Hint

Compare per-thread execution context with shared process resources and the stated thread mapping.

Answer A: User-level threads are managed by a user-space library; kernel-level threads are known to and managed by the OS

User-level threads are managed by a library, while kernel threads are execution contexts known to the OS. Their mapping affects blocking behavior and use of multiple CPUs. The management layer alone does not guarantee faster creation, faster execution or greater stability; those depend on implementation. It also does not grant user code privileged hardware access. The option keeps the supported management distinction.

Why each choice is right or wrong

A - Correct: This states library management versus OS-known kernel threads.

B - Incorrect: No management layer guarantees that kernel threads are always faster.

C - Incorrect: User-level thread management does not grant direct privileged hardware access.

D - Incorrect: Blocking and CPU-use differences show that the distinction has practical effects.

Primary note proof

Study-note section OS08

> User-level threads are managed by a library; kernel threads are known to the OS. In many-to-one mapping, a blocking kernel call can block all user threads and one kernel execution context cannot use multiple CPUs simultaneously. One-to-one permits another kernel thread to run when one blocks. Many-to-many multiplexes user threads over several kernel threads. Speed and reliability depend on implementation; the management layer alone guarantees neither. Application names such as browser or Excel do not prove a particular version’s threading model.

Study-note section OS-S02

> Being scheduled or managed by the kernel does not give application thread code kernel privilege. Ordinary applications and supporting utilities execute in restricted user mode and request protected services through libraries and system calls. Kernel component code executes in privileged kernel mode and performs permitted low-level hardware operations.

Question sources

Operating Systems review questions · Question 31

Question 215 | NS0715

What is IPC (Inter-Process Communication)?

A. A CPU scheduling algorithm

B. Mechanisms allowing processes to exchange information, synchronize, and coordinate activities

C. A method for allocating memory between processes

D. A file system operation

Hint

Identify how data moves, which participant waits and what event allows the operation to continue.

Answer B: Mechanisms for exchanging information and coordinating processes.

OS09 describes shared-memory communication, send/receive, pipes, queues, sockets and RPC, including synchronization for conflicting updates. B summarizes the exchange and coordination role. IPC is not a CPU selection algorithm or a particular memory-allocation/file operation, even though shared memory and file-based mechanisms may be used to implement it.

Why each choice is right or wrong

A - Incorrect: CPU scheduling selects execution opportunities, rather than defining communication and coordination among processes.

B - Correct: Exchanging information with appropriate synchronization/coordination captures the IPC mechanisms described here.

C - Incorrect: Shared memory can implement IPC, but memory allocation alone does not define interprocess communication.

D - Incorrect: Files can participate in communication, but IPC is not restricted to one filesystem operation.

Primary note proof

Study-note section OS09

> Shared memory lets processes access a common region and requires synchronization for conflicting updates. Message passing exchanges data through send/receive operations. Pipes provide streams; message queues preserve messages; sockets connect endpoints; RPC requests a remote operation. HTTP is an application protocol over a communication mechanism and can carry interprocess communication, so it is not universally “not IPC.”

Study-note section OS09

File locks coordinate access rather than transfer arbitrary payloads, though locking is synchronization within IPC.

Question sources

Operating Systems review questions · Question 32

Question 216 | NS0716

What is the Critical Section problem concerned with?

A. Preventing processes from using too much CPU

B. Ensuring that no two processes are executing their critical sections (code accessing shared resources) simultaneously

C. Managing disk I/O operations

D. Scheduling processes in real-time systems

Hint

Identify conflicting shared accesses and the safety or progress property required.

Answer B: Mutually exclude conflicting critical-region execution.

OS10 defines a critical section as code accessing shared state needing protection and lists mutual exclusion, progress and bounded waiting as requirements. B states the exclusion aspect for critical sections protecting the same conflicting resource. The problem also includes progress/fair-waiting requirements, so exclusion is not its entire specification. CPU quotas, disk management and real-time scheduling are different topics.

Why each choice is right or wrong

A - Incorrect: CPU consumption limits are not the shared-access exclusion problem defining critical sections.

B - Correct: Conflicting critical regions must not execute simultaneously; this is the mutual-exclusion part of the correctness requirements.

C - Incorrect: Disk I/O can involve shared state, but managing it generally is not the critical-section problem definition.

D - Incorrect: Real-time selection addresses timing/scheduling, not the core shared-state exclusion requirement requested.

Primary note proof

Study-note section OS10

A race occurs when results depend on uncoordinated interleaving of conflicting accesses. A critical section is code accessing shared state that needs protection, not the resource itself. A correct exclusion solution requires mutual exclusion, progress and bounded waiting under stated assumptions. Mutual exclusion is a desired safety property; races, starvation and deadlock are possible failures.

Study-note section OS10

Keep critical regions small and ensure every conflicting access uses the same protection. A barrier waits until a defined set of participants reaches a point; a monitor encapsulates protected operations with condition synchronization. Locking can itself introduce deadlock if acquisition order is inconsistent.

Question sources

Operating Systems review questions · Question 33

Question 217 | NS0717

For a general process that may perform I/O, which formula defines turnaround time from event timestamps? Waiting means time in the ready queue; burst means total CPU service. The formula must remain valid for nonzero arrival times and blocked intervals.

A. Burst Time - Waiting Time

B. Completion Time - Arrival Time

C. Start Time + Burst Time

D. Waiting Time + Burst Time

Hint

Construct a timeline and distinguish completion rates from individual elapsed-time measures.

Answer B: Completion Time - Arrival Time

Turnaround is completion time minus arrival time, including CPU execution, ready-queue waiting and blocked intervals. Ready waiting plus CPU service omits blocked time when I/O occurs. Start plus burst may resemble a completion timestamp for uninterrupted work, but it is not elapsed turnaround and does not subtract arrival. Burst minus waiting has no general elapsed-time interpretation. Only B remains valid in the full stated model.

Why each choice is right or wrong

A - Incorrect: Subtracting ready waiting from CPU service is not turnaround.

B - Correct: Completion minus arrival includes the entire elapsed lifetime.

C - Incorrect: Start plus burst is not generally completion with I/O and does not subtract arrival.

D - Incorrect: Ready waiting plus CPU service omits blocked intervals.

Primary note proof

Study-note section OS13

> For arrival A, first start S and completion C, response=S−A and turnaround=C−A. For a single CPU burst B with no I/O, waiting=C−A−B. With I/O, subtract all CPU and blocked time to isolate ready-queue waiting.

Question sources

Operating Systems review questions · Question 34

Question 218 | NS0718

What is the Convoy Effect in FCFS scheduling?

A. High-priority processes always complete first

B. All processes receive equal CPU time

C. Short processes get stuck waiting behind one very long process, causing high average waiting times

D. Multiple processes share the same memory space

Hint

Follow arrival order, include idle periods and distinguish waiting from completion time.

Answer C: Short jobs wait behind a long FCFS burst.

OS15 describes the convoy effect as long CPU-bound work making many short jobs wait under non-preemptive FCFS. C directly states that queueing effect and its high waiting-time consequence. FCFS does not use priority order or grant equal total CPU, and shared address space is unrelated to the arrival-order delay. The effect depends on a long burst preceding shorter ones.

Why each choice is right or wrong

A - Incorrect: Priority order does not define FCFS, which follows arrival order rather than high-priority completion.

B - Incorrect: FCFS serves entire bursts of possibly different lengths rather than enforcing equal CPU shares.

C - Correct: Short jobs are delayed behind the long non-preempted burst, producing the FCFS convoy effect.

D - Incorrect: Shared memory concerns address-space organization rather than the scheduling delay called a convoy.

Primary note proof

Study-note section OS15

> FCFS runs ready bursts in arrival order and ordinarily does not preempt. If the CPU is idle, jump to the next arrival. Long CPU-bound work can make many short jobs wait: the convoy effect.

Study-note section OS18

> RR gives recurring opportunities, not identical total CPU consumption for jobs of different lengths.

Study-note section OS08

> A thread has its own PC, registers and stack. Threads in a process share code, address space and usually open-file resources.

Question sources

Operating Systems review questions · Question 35

Question 219 | NS0719

What is Cooperative (Non-Preemptive) vs Preemptive Scheduling?

A. They are the same thing

B. Non-preemptive: process runs until it voluntarily releases CPU; Preemptive: OS can forcibly take CPU from a running process for another

C. Preemptive scheduling never interrupts processes

D. Non-preemptive scheduling always gives better performance

Hint

Check whether the running burst can be interrupted before completion or blocking.

Answer B: Voluntary release versus possible forced CPU preemption.

OS14 says non-preemptive scheduling retains the CPU until completion, blocking or voluntary yielding, while preemptive scheduling may return running work to ready. B captures the voluntary-versus-forced distinction, with completion/blocking included among natural release events. C contradicts preemption, D claims an unsupported universal performance ranking, and A erases the defining difference.

Why each choice is right or wrong

A - Incorrect: The policies differ in whether a running CPU burst can be forcibly interrupted for another process.

B - Correct: Non-preemptive work releases naturally/voluntarily; preemptive policy may forcibly reassign CPU execution time.

C - Incorrect: Forcibly interrupting running work is exactly what preemptive scheduling permits.

D - Incorrect: No policy universally performs better for all workloads and objectives, so the absolute performance claim fails.

Primary note proof

Study-note section OS14

> Non-preemptive scheduling keeps a running CPU burst until completion, blocking or voluntary yielding. Preemptive scheduling can return running work to ready, usually after a timer quantum or a more urgent arrival. FCFS and non-preemptive SJF are non-preemptive; RR and SRTF are preemptive. Priority can be either, so its name alone cannot decide. Cooperative describes relying on voluntary release.

Study-note section OS21

> There is no universally best policy for response, waiting, fairness and throughput simultaneously.

Question sources

Operating Systems review questions · Question 36

Question 220 | NS0720

Which scheduling criterion measures the number of processes completed per unit of time?

A. CPU Utilization

B. Response Time

C. Throughput

D. Turnaround Time

Hint

Construct a timeline and distinguish completion rates from individual elapsed-time measures.

Answer C: Throughput.

OS13 defines throughput as completed jobs divided by elapsed time. That exactly matches the stem's process-completion rate. CPU utilization is busy-time fraction; response measures delay to first start, and turnaround delay to completion. The units therefore distinguish throughput from the other criteria without requiring a scheduling example.

Why each choice is right or wrong

A - Incorrect: CPU utilization measures busy-time fraction, not how many processes finish per unit time.

B - Incorrect: Response measures first-service delay for a process rather than a completion-count rate.

C - Correct: Throughput is explicitly the number of completed jobs divided by elapsed time.

D - Incorrect: Turnaround measures arrival-to-completion elapsed duration rather than completed jobs per time.

Primary note proof

Study-note section OS13

> For arrival A, first start S and completion C, response=S−A and turnaround=C−A. For a single CPU burst B with no I/O, waiting=C−A−B. With I/O, subtract all CPU and blocked time to isolate ready-queue waiting. Throughput=completed jobs/elapsed time; CPU utilization=busy time/elapsed time. Minimize response/turnaround/waiting under the selected goal and maximize useful throughput/utilization.

Question sources

Operating Systems review questions · Question 37

Question 221 | NS0721

What does the Job Queue contain?

A. Only processes currently running on the CPU

B. All processes currently in the system, including those not yet admitted to memory

C. Only processes waiting for I/O devices

D. Processes that have already terminated

Hint

Decide whether the process is eligible, running or waiting, then identify the responsible queue or scheduling role.

Answer B: All processes in the broad textbook system/job-queue sense.

OS06 explicitly defines the broad textbook job queue as processes in the system and distinguishes a narrower admission queue awaiting memory. B matches that broad usage, including not-yet-admitted jobs. Ready/running or I/O-blocked work are subsets, not the whole category; already-ended processes are not the intended current job population. The answer retains the note's terminology qualification.

Why each choice is right or wrong

A - Incorrect: Currently running work is only a subset, not the broad population represented by the textbook job queue.

B - Correct: The broad textbook usage includes system processes/jobs, including those awaiting memory admission.

C - Incorrect: I/O waiting belongs to device queues and does not exhaust the broad job-queue population.

D - Incorrect: Already-terminated work is not the current job population intended here's broad system queue.

Primary note proof

Study-note section OS06

> The ready queue contains eligible work; device queues hold blocked requests. The broad textbook job queue contains processes in the system, while an admission queue more narrowly holds jobs awaiting memory/admission. Long-term scheduling admits jobs; medium-term scheduling suspends/swaps them; short-term scheduling selects ready work.

Study-note section OS06

> New processes are created/admitted; ready means eligible for CPU but not currently running. Running uses a CPU. Waiting/blocked means unable to proceed until an event; terminated means execution has ended. Running→ready follows time-slice expiry or another preemption. Running→waiting follows blocking I/O. I/O completion normally gives waiting→ready, not a guaranteed immediate running state.

Question sources

Operating Systems review questions · Question 38

Question 222 | NS0722

In an open seekable file’s state, what does its current file-offset field (the file-position pointer, rather than the descriptor handle) track?

A. The physical location of the file on disk

B. The current position within an open file for the next read or write operation

C. The owner's access permissions

D. The file's creation timestamp

Hint

Distinguish a file operation, a resource handle, an offset and file metadata.

Answer B: The current position within an open file for the next read or write operation

The current offset records the logical position used for the next read or write on a seekable open file. A descriptor is instead a process-local integer handle referring to open-file state. Physical disk-block locations, permissions and creation timestamps describe other mapping or metadata information. By asking specifically about the position field, the revised stem makes B precise without conflating a descriptor with an offset.

Why each choice is right or wrong

A - Incorrect: Physical disk location is separate from the current logical I/O offset.

B - Correct: The file-offset field records the position for the next read or write.

C - Incorrect: Owner permissions are access-control metadata rather than the current position.

D - Incorrect: Creation time is metadata rather than the current I/O offset.

Primary note proof

Study-note section OS36

> A file has content and metadata such as name, size, owner, permissions and timestamps. open obtains an open-file reference, read/write transfer bytes, seek changes an offset, close releases a reference, truncate changes size and unlink removes a name. A file descriptor is a process-local integer handle; the associated open-file state contains the current offset and access mode. Descriptor and offset are not synonyms.

Study-note section OS36

> Read advances the offset on seekable files and may return fewer bytes than requested; zero normally indicates end-of-file.

Question sources

Operating Systems review questions · Question 39

Question 223 | NS0723

What is the purpose of an Access Control List (ACL) in file protection?

A. To track disk space usage

B. To specify exactly which users/groups have which types of access (read, write, execute) to each file

C. To manage disk fragmentation

D. To schedule file I/O operations

Hint

Distinguish identity checks, permission checks and the particular security property at issue.

Answer B: Specify principals and permitted access rights.

OS39 defines an ACL as listing principals and rights. B states that users/groups can receive different read/write/execute permissions for a file. Authentication verifies identity first; authorization uses these permitted operations. Space accounting, fragmentation and I/O ordering do not define an access-control list. The answer does not assume universal default rights or that every filesystem uses exactly the same permission model.

Why each choice is right or wrong

A - Incorrect: Disk-space usage is storage accounting, not the list of principals and allowed operations.

B - Correct: An ACL associates users/groups or other principals with specified access rights on the resource.

C - Incorrect: Fragmentation concerns block placement and free-space layout rather than authorization entries.

D - Incorrect: Scheduling I/O requests controls service order, not which principal is authorized to access a file.

Primary note proof

Study-note section OS39

> Authentication verifies identity; authorization checks permitted operations. An ACL lists principals and rights; owner/group/other bits are a simpler policy model, not universal defaults granting all owners full access or everyone read. Read, write, execute, append and delete are different rights, with filesystem-specific rules.

Study-note section OS29

> Internal fragmentation is unused space inside an allocated unit. External fragmentation is sufficient total free space split into holes that cannot satisfy a required contiguous allocation.

Study-note section OS06

> The ready queue contains eligible work; device queues hold blocked requests.

Question sources

Operating Systems review questions · Question 40

Question 224 | NS0724

What is Contiguous Allocation of disk space?

A. Files are stored in scattered blocks linked by pointers

B. Each file occupies a continuous set of consecutive disk blocks, enabling fast sequential and direct access

C. Files are indexed using a separate index block

D. Files are allocated in fixed-size frames

Hint

Trace how logical file blocks reach physical blocks and account for pointer metadata.

Answer B: A file occupies consecutive disk blocks.

OS38 defines contiguous allocation through consecutive blocks, with logical block i at start+i. This permits sequential and direct access without following a chain. B matches. Scattered pointers describe linked allocation, a separate index describes indexed allocation, and fixed physical memory frames belong to paging rather than file-contiguity organization. Growth and external fragmentation remain drawbacks.

Why each choice is right or wrong

A - Incorrect: Scattered blocks linked by pointers describe linked allocation rather than one consecutive run.

B - Correct: A consecutive block run supports start-plus-index lookup and sequential/direct file access.

C - Incorrect: A separate block-pointer index describes indexed allocation, not contiguous placement itself.

D - Incorrect: Physical memory frames are paging units rather than the disk-block contiguity rule for a file.

Primary note proof

Study-note section OS38

> Contiguous allocation uses consecutive blocks; logical block i is start+i and supports direct access, but growth and external fragmentation are problems. Extents allow several contiguous runs. Linked allocation chains blocks and supports growth without reserving a complete contiguous file; locating block i requires traversing preceding links. FAT stores chains in a table; caching that table does not change the chain into a direct arithmetic mapping.

Study-note section OS38

> Indexed allocation uses a per-file array/tree of block pointers and supports direct lookup with metadata overhead.

Study-note section OS30

> Pages and physical frames have equal fixed size P.

Question sources

Operating Systems review questions · Question 41

Question 225 | NS0725

What problem does Linked Allocation solve compared to Contiguous Allocation?

A. It speeds up random access

B. It eliminates external fragmentation and pre-known file size requirements since blocks can be scattered and linked anywhere

C. It reduces internal fragmentation

D. It requires less memory for storing file metadata

Hint

Trace how logical file blocks reach physical blocks and account for pointer metadata.

Answer B: Linked blocks permit growth without reserving a full contiguous run.

OS38 says linked allocation supports growth without reserving a complete contiguous file. Since successive blocks can occupy separate free locations, it removes the external-contiguity problem and the need to reserve the final length beforehand. B captures that advantage. Random access still requires traversing earlier links, internal last-block waste may remain, and no universal smaller-metadata claim follows.

Why each choice is right or wrong

A - Incorrect: Locating a later linked block requires traversal, so faster random access is not its general advantage.

B - Correct: Separate linked blocks allow incremental growth without a single reserved contiguous region or preknown final length.

C - Incorrect: A partially used final block can still waste space internally; linking does not specifically remove that waste.

D - Incorrect: Pointers add metadata, and the note gives no universal less-memory guarantee over contiguous allocation.

Primary note proof

Study-note section OS38

> Contiguous allocation uses consecutive blocks; logical block i is start+i and supports direct access, but growth and external fragmentation are problems. Extents allow several contiguous runs.

Linked allocation chains blocks and supports growth without reserving a complete contiguous file; locating block i requires traversing preceding links. FAT stores chains in a table; caching that table does not change the chain into a direct arithmetic mapping.

Study-note section OS29

Internal fragmentation is unused space inside an allocated unit. External fragmentation is sufficient total free space split into holes that cannot satisfy a required contiguous allocation.

Question sources

Operating Systems review questions · Question 42

Question 226 | NS0726

In Indexed Allocation, what is the Index Block?

A. A pointer stored in each data block to the next block

B. A dedicated block containing all the disk-block addresses (pointers to data blocks) for a specific file

C. A table stored at the beginning of the disk for all files

D. A block that stores file metadata like size and timestamps

Hint

Trace how logical file blocks reach physical blocks and account for pointer metadata.

Answer B: A per-file block of data-block pointers in the simple indexed model.

OS38 defines indexed allocation through a per-file array/tree of block pointers and gives an index-block capacity example. B describes the simple single-index-block form. Large files may need multiple index levels, so all addresses in one block is model-dependent rather than universal. A is linked allocation, C a global-table description, and D ordinary metadata without the defining data-block pointers.

Why each choice is right or wrong

A - Incorrect: A next pointer in each data block describes linked allocation rather than a separate per-file index.

B - Correct: In the simple indexed model, a dedicated per-file index contains pointers to the file's data blocks.

C - Incorrect: A global table for all files is not the per-file index block described by indexed allocation.

D - Incorrect: Size/timestamps alone do not provide the data-block address index that defines this allocation structure.

Primary note proof

Study-note section OS38

Indexed allocation uses a per-file array/tree of block pointers and supports direct lookup with metadata overhead. A Unix inode can combine direct pointers with indirect levels. Check pointer size, block size, index capacity and whether metadata reads are cached.

Study-note section OS38

A 1024-byte index block with 4-byte pointers holds 256 block addresses. With 1024-byte data blocks, one such index level addresses 256 KiB of file data, excluding metadata and any reserved entries.

Study-note section OS38

FAT stores chains in a table; caching that table does not change the chain into a direct arithmetic mapping.

Question sources

Operating Systems review questions · Question 43

Question 227 | NS0727

What is the difference between Sequential Access and Direct Access for files?

A. They are identical access methods

B. Sequential access reads/writes records in order; direct access allows jumping to any block/record position by specifying its block number

C. Direct access is only possible for text files

D. Sequential access requires more disk space

Hint

Trace how logical file blocks reach physical blocks and account for pointer metadata.

Answer B: Sequential access proceeds in order; direct access selects a position.

OS38 contrasts linked traversal of preceding blocks with contiguous start+i and indexed direct lookup. These illustrate the access distinction in B: sequential progression versus selecting a numbered position without reading every preceding record. Nothing restricts direct access to text or requires sequential files to occupy more disk space merely because of their access method.

Why each choice is right or wrong

A - Incorrect: Order-dependent traversal and direct selection are different access behaviors, not identical methods.

B - Correct: Sequential access advances through records; direct access identifies a chosen block/position for access.

C - Incorrect: The direct block-lookup rule does not impose a text-file-only restriction.

D - Incorrect: Access order alone does not determine a greater storage-space requirement for sequential files.

Primary note proof

Study-note section OS38

> Contiguous allocation uses consecutive blocks; logical block i is start+i and supports direct access, but growth and external fragmentation are problems. Extents allow several contiguous runs. Linked allocation chains blocks and supports growth without reserving a complete contiguous file; locating block i requires traversing preceding links. FAT stores chains in a table; caching that table does not change the chain into a direct arithmetic mapping.

Study-note section OS38

> Indexed allocation uses a per-file array/tree of block pointers and supports direct lookup with metadata overhead.

Question sources

Operating Systems review questions · Question 44

Question 228 | NS0728

What is a Tree-Structured Directory?

A. A directory where all files are in one flat list

B. A hierarchical directory organization with a root directory from which subdirectories and files branch out, each file having a unique path

C. A directory shared among multiple users

D. A temporary directory deleted on reboot

Hint

Follow namespace structure and distinguish names, underlying files and mounted filesystems.

Answer B: A rooted hierarchy of nested directories and files.

OS37 says a tree allows nested directories and absolute/relative paths. In the pure tree model, each entry has a path from the root, matching B. Flat lists instead describe single-level organization; sharing among users and deletion at reboot do not define tree structure. Links can add graph relationships, but that is an extension beyond the pure tree premise.

Why each choice is right or wrong

A - Incorrect: One flat file list is single-level organization rather than nested branches from a root.

B - Correct: A root with nested subdirectories and paths is the tree hierarchy described here.

C - Incorrect: Multiple users sharing a directory does not by itself define its structural hierarchy.

D - Incorrect: Deletion on reboot is a lifetime policy rather than the rooted directory-tree organization.

Primary note proof

Study-note section OS37

> A single-level directory requires unique names in one namespace. Two-level directories separate users. A tree allows nested directories and absolute/relative paths; links can create shared graphs. Modern general-purpose systems commonly present hierarchical namespaces, but a feature name alone does not prove that every system excludes simpler arrangements.

Question sources

Operating Systems review questions · Question 45

Question 229 | NS0729

What is the difference between Hard Links and Symbolic Links in file systems?

A. Hard links can span different file systems; symbolic links cannot

B. Hard links create multiple directory entries pointing to the same file's data blocks; symbolic links are special files containing the path to another file

C. Symbolic links cannot be deleted

D. Hard links are only supported in Windows

Hint

Follow namespace structure and distinguish names, underlying files and mounted filesystems.

Answer B: Hard links name one file object; symbolic links store a pathname.

A hard link supplies another name for the same underlying file object; a symbolic link stores a pathname. Symbolic links can cross filesystems and may dangle, whereas ordinary hard links stay within a filesystem. Linux's link operation supplies a direct counterexample to the Windows-only claim. unlink can remove a symbolic-link name, so the link is not undeletable. B states the correct distinction.

Why each choice is right or wrong

A - Incorrect: The boundary is reversed: symbolic links may cross filesystems, while ordinary hard links remain within one.

B - Correct: Both hard-link names identify the same underlying file; a symbolic link contains a pathname to resolve.

C - Incorrect: A symbolic-link name can be removed with unlink; removing it does not erase the target's contents.

D - Incorrect: Linux provides hard links through link(), so they are not supported only in Windows.

Primary note proof

Study-note section OS37

> A hard link is another name for the same underlying file object, usually within one filesystem; a symbolic link stores a pathname and can point across filesystems or become dangling. Removing one hard link does not erase content while other names/open references retain it.

Study-note section OS-S06

> Linux provides link() to create another hard-link name for an existing file. The old and new names identify the same file, so hard links are not a Windows-only facility. unlink() removes a filesystem name; when that name is a symbolic link, it removes the link itself. Removing a name and erasing the target's contents are different operations.

Question sources

Operating Systems review questions · Question 46

Question 230 | NS0730

What is File System Mounting?

A. Creating a new file system

B. The process of attaching a file system to a specific location in the existing directory tree, making it accessible

C. Deleting old files to free disk space

D. Formatting a disk partition

Hint

Follow namespace structure and distinguish names, underlying files and mounted filesystems.

Answer B: Attach a filesystem at a mount point.

OS37 explicitly defines mounting as attaching a filesystem at a mount point and distinguishes it from formatting. B states that integration into the existing namespace. Creation/formatting establish a filesystem's on-device structure; deleting files frees content space. None of those is the namespace attachment operation requested by mounting.

Why each choice is right or wrong

A - Incorrect: Creating a new filesystem establishes storage structure rather than attaching it to an existing namespace.

B - Correct: Mounting makes a filesystem accessible at a chosen point in the directory hierarchy.

C - Incorrect: Deleting files changes contents/free space rather than attaching a filesystem to the namespace.

D - Incorrect: Formatting is explicitly distinguished from mounting in the supplied note.

Primary note proof

Study-note section OS37

> NFS provides remote filesystem access; FTP transfers files rather than itself mounting a local filesystem.

Study-note section OS35

> Disk initialization/filesystem creation is OS/tool work, firmware begins boot, and drive firmware may remap bad sectors.

Question sources

Operating Systems review questions · Question 47

Question 231 | NS0731

In the context of deadlock recovery, what is 'Rollback'?

A. Restarting the entire operating system

B. Returning a preempted process to a safe previous state after resources are taken back from it during deadlock resolution

C. Deleting all processes involved in a deadlock

D. Rebooting the computer to free all resources

Hint

Separate handling before allocation from actions taken after detection, and consider the cost of checking.

Answer B: Restore an affected process to a prior checkpoint after reclaiming resources.

OS26 describes reclaiming safely preemptable resources and rolling affected work back to a checkpoint. B states that restoration to a safe prior state after resources are withdrawn. Restarting the entire OS/computer and killing all involved processes are different coarse recovery actions, not the meaning of rollback. Safe checkpoints/restart semantics are required to make withdrawal and restoration coherent.

Why each choice is right or wrong

A - Incorrect: Restarting the whole OS is not the checkpoint restoration of affected work that defines rollback.

B - Correct: Rollback restores an affected process to a previous recoverable state after safe resource reclamation.

C - Incorrect: Terminating processes is a separate recovery tool rather than restoring their earlier execution state.

D - Incorrect: Rebooting frees broad machine state but is not the process-checkpoint restoration definition of rollback.

Primary note proof

Study-note section OS26

> Detection asks whether the current state is deadlocked. Recovery can terminate selected processes or reclaim safely preemptable resources and roll affected work back to a checkpoint. Termination and rollback are both possible recovery tools; choose according to the resource and process conditions.

Question sources

Operating Systems review questions · Question 48

Question 232 | NS0732

What is the purpose of relocation/base and limit registers in the standard contiguous single-process-region memory model? The limit denotes the permitted region length.

A. They speed up CPU scheduling decisions

B. The relocation register stores the process's physical base and the limit its permitted length; hardware checks the logical offset before adding the base

C. They track file system locations on disk

D. They manage virtual memory page tables

Hint

Apply the stated hole-selection or base-and-bound rule with its explicit convention.

Answer B: The relocation register stores the process's physical base and the limit its permitted length; hardware checks the logical offset before adding the base

In the standard base/limit model, the base identifies the process's physical starting address and the limit gives its permitted length. Hardware checks that the logical offset is nonnegative and smaller than the limit, then computes base plus offset. This protects accesses to the assigned region. The registers do not describe the OS base and size, schedule CPU work, track disk files or manage page tables.

Why each choice is right or wrong

A - Incorrect: These registers check memory accesses rather than schedule CPU work.

B - Correct: The option states the process base, length bound and checked translation.

C - Incorrect: Disk-block locations are not the purpose of RAM base/limit registers.

D - Incorrect: Simple contiguous base/limit translation does not manage page-table mappings.

Primary note proof

Study-note section OS28

> A contiguous allocation gives a process one continuous region. In the standard base/limit model, base is that process's physical starting address and limit is its permitted length. Check 0≤logical_offset<limit, then physical=base+offset. The relocation register does not normally hold the OS base.

Study-note section OS30

> Pages and physical frames have equal fixed size P. Divide virtual address v into page=floor(v/P) and offset=v mod P. The page table maps a page to a frame and protection/status flags.

Question sources

Operating Systems review questions · Question 49

Question 233 | NS0733

In Dynamic Linking, what is a key advantage over Static Linking?

A. Programs run faster with dynamic linking

B. Multiple programs share one copy of library code in memory rather than each program embedding its own copy, saving memory

C. Dynamic linking eliminates the need for a page table

D. Dynamically linked programs cannot have bugs

Hint

Track when an address or library reference is resolved and the direction of translation.

Answer B: Share suitable library code pages across programs.

OS27 says dynamic linking resolves external-library references at load/runtime and allows suitable library code pages to be shared, whereas static linking copies selected code into the executable. B captures the potential memory saving. It is not an unconditional speed guarantee, does not remove address translation/page tables and cannot eliminate programming bugs. Only shareable code/pages are implied, not arbitrary writable library state.

Why each choice is right or wrong

A - Incorrect: Dynamic resolution/sharing does not guarantee faster execution for every program or workload.

B - Correct: Several processes can share suitable library code pages rather than each embedding a separate static copy.

C - Incorrect: Library linking does not remove virtual-address translation or its page-table requirements.

D - Incorrect: Choosing dynamic rather than static linking cannot guarantee the absence of programming errors.

Primary note proof

Study-note section OS27

> Dynamic linking resolves references to external libraries at load/runtime, allowing suitable library code pages to be shared; static linking copies selected code into the executable.

Study-note section OS30

> Pages and physical frames have equal fixed size P. Divide virtual address v into page=floor(v/P) and offset=v mod P. The page table maps a page to a frame and protection/status flags.

Question sources

Operating Systems review questions · Question 50