General Questions on Operating Systems

1. Which of the following statements best describes the concept of kernel mode?
A) A mode in which user processes directly access the hardware
B) A mode in which the operating system executes code with elevated privileges
C) A mode used only by application programs
D) A mode reserved for external device drivers
Correct answer: B

Explanation: Kernel mode (or supervisor mode) allows the operating system to execute privileged instructions (e.g., memory management, I/O). User processes, on the other hand, operate in user mode.

2. Which of the following operations is directly managed by the operating system?
A) Source code compilation
B) Access to I/O devices
C) Interpretation of scripting languages
D) Debugging of user code
Correct answer: B

Explanation: Interaction with I/O devices is a typical function of the operating system, carried out through dedicated drivers within the kernel.

3. What is the main purpose of the file system?
A) To provide an interface for accessing files and secondary storage
B) To manage network communication between processes
C) To translate system calls into machine code
D) To execute background processes
Correct answer: A

Explanation: The file system manages the storage, organization, and access to files on the disk, providing a high-level abstraction.

4. During a context switch, which of the following pieces of information must always be saved?
A) Disk cache contents
B) The page table of all processes
C) Shared virtual memory contents
D) The CPU registers of the running process
Correct answer: D

Explanation: During a context switch, the operating system must save the state of the interrupted process, which includes the CPU registers (Program Counter, Stack Pointer, general-purpose registers). This allows the process to resume execution exactly from the point where it was suspended.

Other elements (disk cache, global virtual memory, page tables of all processes) are not part of an individual process's context.

5. What does the `exec()` system call represent in Unix/Linux?
A) It creates a new child process
B) It terminates the calling process
C) It replaces the code of the current process with a new program
D) It suspends the current process until a child terminates
Correct answer: C

Explanation: `exec()` replaces the memory space and code of the calling process with a new program to be executed.

CPU Scheduling Exercises

6. Compute the average waiting time using the First Come First Serve (FCFS) algorithm for the following system:
Processes and CPU times:
P1: Arrival = 0, CPU = 5 ms
P2: Arrival = 0, CPU = 3 ms
P3: Arrival = 0, CPU = 8 ms
P4: Arrival = 0, CPU = 6 ms

A) 5.5 ms
B) 6.25 ms
C) 7.25 ms
D) 7.5 ms
Correct answer: C

Solution:
- Order of execution: P1 → P2 → P3 → P4
- Waiting times
  - wt(P1) = 0;
  - wt(P2) = 5;
  - wt(P3) = 5+3=8;
  - wt(P4) = 5+3+8=16.

  Avg. = (0 + 5 + 8 + 16)/4 = **7.25 ms**

7. Compute the average waiting time using FCFS:
Processes and CPU times:
P1: Arrival = 0, CPU = 4 ms
P2: Arrival = 1, CPU = 6 ms
P3: Arrival = 2, CPU = 3 ms
P4: Arrival = 4, CPU = 5 ms

A) 4.5 ms
B) 5 ms
C) 5.25 ms
D) 6.75 ms
Correct answer: B

Solution:

- Order of execution: P1 → P2 → P3 → P4
- Waiting times
    - wt(P1) = 0;
    - wt(P2) = 4-1 = 3;
    - wt(P3) = 13-3-2 = 8;
    - wt(P4) = 18-5-4 = 9.

  Avg. = (0 + 3 + 8 + 9)/4 = **5 ms**

8. Compute the average waiting time using Round Robin (RR) with time quantum ***q = 2 ms****:*
Processes and CPU times:
P1: Arrival = 0, CPU = 5 ms
P2: Arrival = 0, CPU = 3 ms
P3: Arrival = 0, CPU = 6 ms
P4: Arrival = 0, CPU = 4 ms

A) 9 ms
B) 9.25 ms
C) 10 ms
D) 10.5 ms
Correct answer: D

Solution:

Time Interval	Process	Execution	Remaining Time
0-2	P1	2	3
2-4	P2	2	1
4-6	P3	2	4

6-8	P4	2	2
8-10	P1	2	1
10-**11**	P2	1	0 (finisce)
11-13	P3	2	2
13-**15**	P4	2	0 (finisce)
15-**16**	P1	1	0 (finisce)
16-**18**	P3	2	0 (finisce)

- Order of execution: P1 → P2 → P3 → P4 → P1 → P3 → P4 → P1 → P3
- Waiting times
    - wt(P1) = 16-5 = 11;
    - wt(P2) = 11-3 = 8;
    - wt(P3) = 18-6 = 12;
    - wt(P4) = 15-4 = 11.

  Avg. = (11 + 8 + 12 + 11)/4 = 42/4 = **10.5 ms**

*9. Compute the average waiting time using Round Robin (RR) with **q = 3 ms**:*
Processes and CPU times:
P1: Arrival = 0, CPU = 9 ms
P2: Arrival = 1, CPU = 5 ms
P3: Arrival = 7, CPU = 6 ms
P4: Arrival = 11, CPU = 4 ms

A) 7.5 ms
B) 8 ms
C) 8.25 ms
D) 9 ms
Correct answer: B

Solution:

Time Interval	**Process**	**Execution**	**Remaining Time**
0-3	P1	3	6
3-6	P2	3	2
6-9	P1 (P3 arrives at t=7 behind P2)	3	3

9-**11**	P2	2	0 (terminates)
11-14	P3 (P4 arrives at t=11 behind P1)	3	3
14-**17**	P1	3	0 (terminates)
17-20	P4	3	1
20-**23**	P3	3	0 (terminates)
23-**24**	P4	1	0 (terminates)

- Order of execution: P1 → P2 → P1 → P2 → P3 → P1 → P4 → P3 → P4
- Waiting times
    - wt(P1) = 17-9-0 = 8;
    - wt(P2) = 11-5-1 = 5;
    - wt(P3) = 23-6-7 = 10;
    - wt(P4) = 24-4-11 = 9;

  Avg. = (8 + 5 + 10 + 9)/4 = 32/4 = **8 ms**

10. Compute the average waiting time using Shortest Job First (SJF – non-preemptive):
Processes and CPU times:
P1: Arrival = 0, CPU = 8
P2: Arrival = 0, CPU = 2
P3: Arrival = 0, CPU = 6
P4: Arrival = 0, CPU = 4

A) 4.5 ms
B) 4.75 ms
C) 5 ms
D) 5.5 ms
Correct answer: C

Solution:
- Sorting processes by CPU times: P2 → P4 → P3 → P1
- Waiting times
    - wt(P1) = 20-8 = 12;
    - wt(P2) = 2-2 = 0;
    - wt(P3) = 12-6 = 6;
    - wt(P4) = 6-4 = 2.

  Avg. = (12+0+6+2)/4 = **5 ms**

11. Compute the average waiting time using Shortest Remaining Time First (SRTF – preemptive):

Processes and CPU times:
P1: Arrival = 0, CPU = 8
P2: Arrival = 2, CPU = 5
P3: Arrival = 4, CPU = 2

A) 3 ms
B) 3.33 ms
C) 3.67 ms
D) 4.16 ms
Correct answer: A

<u>**Solution:**</u>

- P1 starts at t=0
- SRTF interrupts P1 at t=2 since its remaining time (8-2=6) is greater than P2 (5)
- SRTF interrupts P2 at t=4 since its remaining time (5-2=3) is greater than P3 (2)
- P3 runs from t=4 to t=6
- P2 resumes its execution at t=6 until t=9
- P1 resumes its execution at t=9 until t=15
- Waiting times
    - wt(P1) = (15-8-0) = 7;
    - wt(P2) = (9-5-2) = 2;
    - wt(P3) = (6-2-4) = 0.

  Avg. = (7+2+0)/3 = 9/3 = **3 ms**

12. Compute the average waiting time using Round Robin (RR) (Arrival = 0 for all, ***q = 4 ms****):*
Processes, CPU and I/O times:
P1: CPU 6, I/O 4, CPU 4
P2: CPU 5, I/O 2, CPU 3
P3: CPU 4, I/O 3, CPU 2

A) 8.33 ms
B) 8.67 ms
C) 9.25 ms
D) 9.33 ms
Correct answer: D

<u>**Solution:**</u>

Time Interval	Process	Execution	Remaining Time
0-4	P1	4	CPU 2, I/O 3, CPU 4
4-8	P2	4	CPU 1, I/O 2, CPU 3

8-12	P3 (I/O from t=12 to t=15)	4	I/O 3, CPU 2
12-14	P1 (I/O from t=14 to t=18)	2	I/O 4, CPU 4
14-15	P2 (I/O from t=15 to t=17)	1	I/O 2, CPU 3
15-**17**	P3	2	0 (terminates)
17-**20**	P2	3	0 (terminates)
20-**24**	P1	4	0 (terminates)

- Order of execution: P1 → P2 → P3 → P1 → P2 → P3 → P2 → P1
- Waiting times
  - wt(P1) = 24-(6+4)-4 = 10;
  - wt(P2) = 20-(5+3)-2 = 10;
  - wt(P3) = 17-(4+2)-3 = 8.

  Avg. = (10 + 10 + 8)/3 ~ **9.33 ms**

Process and Thread Creation

13. In a system with 10 processes, each one executes a `fork()`*. Assuming all calls succeed, how many processes will exist in total?*
A) 10
B) 20
C) 100
D) Depends on the OS
Correct answer: B

<u>Explanation:</u> `fork()` duplicates the calling process → total of 2 processes (parent + child) for each of the 10 processes that calls it, i.e., 2 × 10 = 20.

14. What happens if a process calls `wait()` *when it has no active child processes?*
A) It remains indefinitely blocked
B) It terminates automatically
C) The operating system returns an imAvg.te error
D) It blocks until another process wakes it up
Correct answer: C

<u>Explanation:</u> `wait()` returns an error (-1) and sets `errno=ECHILD`.

15. In a multithreaded process, what do all threads share?
A) Stack
B) Program counter register
C) Address space (memory and global variables)
D) Thread identifier
Correct answer: C

Explanation: Threads share the memory and resources of the process, but each thread has its own stack and program counter (PC).

16. What is the purpose of the `pthread_join()` *call?*
A) To create a new thread
B) To terminate a thread
C) To force thread synchronization
D) To wait for the termination of a specific thread
Correct answer: D

Explanation: The `pthread_join()` function in POSIX Threads (Pthreads) is used to synchronize the calling thread with the termination of a specific thread. The calling thread is blocked until the thread identified by thread has finished execution.

17. Which situation may occur if two threads access a global variable simultaneously without synchronization?
A) Starvation
B) Deadlock
C) Race condition
D) Context switch
Correct answer: C

Explanation: The order of access can lead to non-deterministic results → race condition.

Synchronization (Locks, Semaphores, Monitors)

18. Which property is essential to guarantee mutual exclusion?
A) All processes must be able to enter the critical section simultaneously
B) At most one process can be in the critical section at any given time
C) Each process must enter the critical section within a finite amount of time
D) A process must never be allowed to access the critical section
Correct answer: B

<u>Explanation:</u> Mutual exclusion is the fundamental property of synchronization mechanisms that prevents concurrent access to shared resources. It ensures that only one process or thread at a time can be inside the critical section, which is the block of code that accesses or modifies shared resources.

19. A semaphore initialized to 3 can be used to:
A) Allow up to three threads to access a shared resource simultaneously
B) Allow only one thread at a time to access a shared resource
C) Block all threads indefinitely
D) Implement one-to-one communication between two threads
Correct answer: A

<u>Explanation:</u> A counting semaphore initialized to 3 allows up to 3 simultaneous accesses.

20. Which of the following synchronization techniques avoids busy waiting?
A) Spinlock
B) Test-and-set lock
C) Monitor
D) Preemption
Correct answer: C

<u>Explanation:</u> Busy waiting occurs when a thread continuously loops, checking a condition, thereby consuming CPU resources. Spinlocks and test-and-set locks are classic examples of this behavior, as they repeatedly attempt to acquire the lock. Preemption is not a synchronization technique; it is a scheduler feature that interrupts a process to assign the CPU to another process. Monitors, on the other hand, use mechanisms such as *wait*/*signal* to suspend the thread, putting it to "sleep" until the resource becomes available, thus avoiding busy waiting.