Domande Generali sui Sistemi Operativi

1. Quale delle seguenti affermazioni descrive meglio il concetto di "kernel mode"?
A) Una modalità in cui i processi utente accedono direttamente all'hardware
B) Una modalità in cui il sistema operativo esegue codice con privilegi elevati
C) Una modalità usata solo dai programmi applicativi
D) Una modalità riservata ai driver di periferica esterni
Risposta corretta: B

Spiegazione: Il *kernel mode* (o *supervisor mode*) consente al sistema operativo di eseguire istruzioni privilegiate (es. gestione memoria, I/O). I processi utente operano invece in user mode.

2. Quale delle seguenti operazioni è gestita direttamente dal sistema operativo?
A) Compilazione del codice sorgente
B) Accesso ai dispositivi di I/O
C) Interpretazione di linguaggi di scripting
D) Debug del codice utente
Risposta corretta: B

Spiegazione: L'interazione con i dispositivi di I/O è una funzione tipica del sistema operativo, tramite appositi driver presenti nel kernel.

3. Qual è lo scopo principale del file system?
A) Fornire un'interfaccia per l'accesso ai file e alla memoria secondaria
B) Gestire la comunicazione di rete tra processi
C) Tradurre le chiamate di sistema in codice macchina
D) Eseguire i processi in background
Risposta corretta: A

Spiegazione: Il file system gestisce la memorizzazione, l'organizzazione e l'accesso ai file nel disco, fornendo un'astrazione di alto livello.

4. Durante un context switch, quale delle seguenti informazioni deve essere sempre salvata?
A) Il contenuto della cache del disco
B) La tabella delle pagine di tutti i processi
C) Il contenuto della memoria virtuale condivisa
D) Il contenuto dei registri della CPU del processo in esecuzione
Risposta corretta: D

Spiegazione: Durante un *context switch* il sistema operativo deve salvare lo stato del processo interrotto, che include i registri della CPU (Program Counter, Stack Pointer, registri generali). Questo permette di riprendere l'esecuzione del processo esattamente dal punto in cui era stato sospeso.

Gli altri elementi (cache disco, memoria virtuale globale, tabelle di tutti i processi) non sono parte del contesto individuale.

5. Cosa rappresenta la system call `exec()` *in Unix/Linux?*
A) Crea un nuovo processo figlio
B) Termina il processo chiamante
C) Sostituisce il codice del processo corrente con un nuovo programma
D) Sospende il processo corrente fino alla terminazione di un figlio
Risposta corretta: C

Spiegazione: `exec()` rimpiazza lo spazio di memoria e il codice del processo chiamante con un nuovo programma da eseguire.

Esercizi su Scheduling della CPU

6. Calcolare il tempo medio di attesa con l'algoritmo First Come First Serve (FCFS) del seguente sistema:
Processi e tempi di esecuzione:
P1: Arrivo = 0, Esecuzione = 5 ms
P2: Arrivo = 0, Esecuzione = 3 ms
P3: Arrivo = 0, Esecuzione = 8 ms
P4: Arrivo = 0, Esecuzione = 6 ms

A) 5.5 ms
B) 6.25 ms
C) 7.25 ms
D) 7.5 ms
Risposta corretta: C

Soluzione:

- Ordine di esecuzione: P1 → P2 → P3 → P4
- Tempi di attesa
    - wt(P1) = 0;
    - wt(P2) = 5;
    - wt(P3) = 5+3=8;
    - wt(P4) = 5+3+8=16.

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

7. Calcolare il tempo medio di attesa con l'algoritmo First Come First Serve (FCFS) del seguente sistema:
Processi e tempi di esecuzione:
P1: Arrivo = 0, Esecuzione = 4 ms
P2: Arrivo = 1, Esecuzione = 6 ms
P3: Arrivo = 2, Esecuzione = 3 ms
P4: Arrivo = 4, Esecuzione = 5 ms

A) 4.5 ms
B) 5 ms
C) 5.25 ms
D) 6.75 ms
Risposta corretta: B

Soluzione:

- Ordine di esecuzione: P1 → P2 → P3 → P4
- Tempi di attesa
    - wt(P1) = 0;
    - wt(P2) = 4-1 = 3;
    - wt(P3) = 13-3-2 = 8;
    - wt(P4) = 18-5-4 = 9.

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

8. Calcolare il tempo medio di attesa con l'algoritmo Round Robin (RR) del seguente sistema:
Processi e tempi di esecuzione (q = 2 ms):
P1: Arrivo = 0, Esecuzione = 5 ms
P2: Arrivo = 0, Esecuzione = 3 ms
P3: Arrivo = 0, Esecuzione = 6 ms
P4: Arrivo = 0, Esecuzione = 4 ms

A) 9 ms
B) 9.25 ms
C) 10 ms
D) 10.5 ms
Risposta corretta: D

Soluzione:

Intervallo	Processo	Esecuzione	Rimanente
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)

- Ordine di esecuzione: P1 → P2 → P3 → P4 → P1 → P3 → P4 → P1 → P3
- Tempi di attesa
    - wt(P1) = 16-5 = 11;
    - wt(P2) = 11-3 = 8;
    - wt(P3) = 18-6 = 12;
    - wt(P4) = 15-4 = 11.

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

9. Calcolare il tempo medio di attesa con l'algoritmo Round Robin (RR) del seguente sistema:
Processi e tempi di esecuzione (q = 3 ms):
P1: Arrivo = 0, Esecuzione = 9
P2: Arrivo = 1, Esecuzione = 5
P3: Arrivo = 7, Esecuzione = 6
P4: Arrivo = 11, Esecuzione = 4

A) 7.5 ms
B) 8 ms
C) 8.25 ms
D) 9 ms
Risposta corretta: B

Soluzione:

Intervallo	Processo	Esecuzione	Rimanente
0-3	P1	3	6
3-6	P2	3	2

6-9	P1 (P3 arriva a t=7 e si accoda a P2)	3	3
9-**11**	P2	2	0 (finisce)
11-14	P3 (P4 arriva a t=11 e si accoda a P1)	3	3
14-**17**	P1	3	0 (finisce)
17-20	P4	3	1
20-**23**	P3	3	0 (finisce)
23-**24**	P4	1	0 (finisce)

- Ordine di esecuzione: P1 → P2 → P1 → P2 → P3 → P1 → P4 → P3 → P4
- Tempi di attesa
    - wt(P1) = 17-9-0 = 8;
    - wt(P2) = 11-5-1 = 5;
    - wt(P3) = 23-6-7 = 10;
    - wt(P4) = 24-4-11 = 9;

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

10. Calcolare il tempo medio di attesa con l'algoritmo Shortest Job First (SJF - non preemptive):
Processi e tempi di esecuzione:
P1: Arrivo = 0, CPU = 8
P2: Arrivo = 0, CPU = 2
P3: Arrivo = 0, CPU = 6
P4: Arrivo = 0, CPU = 4

A) 4.5 ms
B) 4.75 ms
C) 5 ms
D) 5.5 ms
Risposta corretta: C

<u>Soluzione:</u>

- Ordine di esecuzione per durata: P2 → P4 → P3 → P1
- Tempi di attesa
    - wt(P1) = 20-8 = 12;
    - wt(P2) = 2-2 = 0;
    - wt(P3) = 12-6 = 6;
    - wt(P4) = 6-4 = 2.

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

11. Calcolare il tempo medio di attesa con l'algoritmo Shortest Remaining Time First (SRTF - preemptive):
Processi e tempi di esecuzione:
P1: Arrivo = 0, CPU = 8
P2: Arrivo = 2, CPU = 5
P3: Arrivo = 4, CPU = 2

A) 3 ms
B) 3.33 ms
C) 3.67 ms
D) 4.16 ms
Risposta corretta: A

Soluzione:

- P1 inizia a t=0
- SRTF interrompe P1 a t=2 perché il suo tempo rimanente (8-2=6) è maggiore del tempo di P2 (5)
- SRTF interrompe P2 a t=4 perché il suo tempo rimanente (5-2=3) è maggiore del tempo di P3 (2)
- P3 esegue da t=4 a t=6
- P2 riprende l'esecuzione da t=6 a t=9
- P1 riprende l'esecuzione da t=9 a t=15
- Tempi di attesa
    - wt(P1) = (15-8-0) = 7;
    - wt(P2) = (9-5-2) = 2;
    - wt(P3) = (6-2-4) = 0.

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

12. Calcolare il tempo medio di attesa con l'algoritmo Round Robin (RR) del seguente sistema:
Processi, tempi di esecuzione e di I/O (arrivo = 0 per tutti; q = 4 ms):
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
Risposta corretta: D

Soluzione:

Intervallo	Processo	Esecuzione	Rimanente

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 da t=12 a t=15)	4	I/O 3, CPU 2
12-14	P1 (I/O da t=14 a t=18)	2	I/O 4, CPU 4
14-15	P2 (I/O da t=15 a t=17)	1	I/O 2, CPU 3
15-**17**	P3	2	0 (finisce)
17-**20**	P2	3	0 (finisce)
20-**24**	P1	4	0 (finisce)

- Ordine di esecuzione: P1 → P2 → P3 → P1 → P2 → P3 → P2 → P1
- Tempi di attesa
  - wt(P1) = 24-(6+4)-4 = 10;
  - wt(P2) = 20-(5+3)-2 = 10;
  - wt(P3) = 17-(4+2)-3 = 8.

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

Creazione di Processi e Thread

13. In un sistema con 10 processi, ciascuno di essi esegue una `fork()`*. Supponendo che tutte le chiamate vadano a buon fine, quanti processi saranno presenti in totale nel sistema?*
A) 10
B) 20
C) 100
D) Dipende dal sistema operativo
Risposta corretta: B

Spiegazione: `fork()` duplica il processo chiamante → totale 2 processi (padre + figlio) per ciascuno dei 10 che esegue la chiamata, ossia 2 * 10 = 20.

14. Cosa accade se un processo chiama `wait()` *senza figli attivi?*
A) Rimane in attesa indefinita
B) Termina automaticamente
C) Il sistema operativo genera un errore immediato
D) Si blocca finché un altro processo lo risveglia

Risposta corretta: C

Spiegazione: `wait()` ritorna con errore (-1) e imposta `errno=ECHILD`.

15. In un processo multithreaded, cosa condividono tutti i thread?
A) Stack
B) Registro program counter
C) Spazio di indirizzamento (memoria e variabili globali)
D) Identificatore di thread
Risposta corretta: C

Spiegazione: I thread condividono memoria e risorse del processo, ma ciascuno ha il proprio stack e PC.

16. Qual è lo scopo della chiamata `pthread_join()`*?*
A) Creare un nuovo thread
B) Terminare un thread
C) Forzare la sincronizzazione dei thread
D) Attendere la terminazione di un thread specifico
Risposta corretta: D

Spiegazione: La funzione `pthread_join()` in POSIX Threads (Pthreads) serve a sincronizzare il thread chiamante con la terminazione di un thread specifico. Il thread chiamante viene bloccato finché il thread identificato da thread non termina.

17. Quale situazione può verificarsi se due thread accedono contemporaneamente a una variabile globale senza sincronizzazione?
A) Starvation
B) Deadlock
C) Race condition
D) Context switch
Risposta corretta: C

Spiegazione: L'ordine di accesso determina risultati non deterministici → race condition.

Sincronizzazione (Lock, Semafori, Monitor)

18. Quale proprietà è essenziale per garantire la mutual exclusion?
A) Tutti i processi devono poter entrare contemporaneamente nella sezione critica
B) Al massimo un processo per volta può trovarsi nella sezione critica
C) Ogni processo deve entrare nella sezione critica entro un tempo finito
D) Un processo non deve mai poter accedere alla sezione critica

Risposta corretta: B

Spiegazione: La *mutual exclusion* è la proprietà fondamentale dei meccanismi di sincronizzazione che serve a prevenire l'accesso concorrente a risorse condivise. Significa che solo un processo o thread per volta può trovarsi all'interno della sezione critica, ovvero il blocco di codice che accede o modifica risorse condivise.

19. Un semaforo con valore iniziale 3 può essere usato per:
A) Consentire a tre thread per volta di accedere contemporaneamente a una risorsa condivisa
B) Consentire a un solo thread per volta di accedere a una risorsa condivisa
C) Bloccare indefinitamente tutti i thread in attesa
D) Implementare una comunicazione one-to-one tra due thread
Risposta corretta: A

Spiegazione: Un counting semaphore inizializzato a 3 consente fino a 3 accessi simultanei.

20. Quale delle seguenti tecniche di sincronizzazione evita il busy waiting?
A) Spinlock
B) Test-and-set lock
C) Monitor
D) Preemption
Risposta corretta: C

Spiegazione: Il *busy waiting* (o "attesa attiva") si verifica quando un thread resta in un ciclo continuo "vuoto" per verificare una condizione, consumando CPU.
Le *spinlock* e le *test-and-set lock* ne sono esempi classici, perché fanno proprio questo.
La *preemption* non è una tecnica di sincronizzazione, ma una funzionalità dello scheduler che interrompe un processo per assegnare la CPU a un altro.
I *monitor*, invece, usano meccanismi di sospensione del thread (*wait*/*signal*) che lo mettono in stato di "sleep" finché la risorsa non è disponibile, evitando così il busy waiting.