

Simulazione prova pratica

- Scheduling
- Memoria
- Processi

Esercizio 1

- Supponiamo che nella coda di attesa siano presenti i seguenti processi:

<i>Processo</i>	<i>CPU burst</i>	<i>Arrivo</i>	<i>Priorità</i>
P1	10	0	3
P2	6	2	5
P3	4	6	2
P4	4	7	1
P5	6	8	4

- Disegnare il diagramma di Gantt e calcolare il tempo medio di turnaround (completamento) per le politiche di scheduling:
 - Round Robin con quanto 2,**
 - SJF con prelazione,**
 - FCFS e**
 - Priorità.**

Soluzione

(a) RR:

- ◆ $P_1(0:2) P_2(2:4) P_1(4:6) P_2(6:8) P_3(8:10) P_1(10:12) P_4(12:14) P_5(14:16) P_2(16:18)$
 $P_3(18:20) P_1(20:22) P_4(22:24) P_5(24:26) P_1(26:28) P_5(28:30)$
- ◆ $T = (28+(18-2)+(20-6)+(24-7)+(30-8))/5 = 19,4$

(b) SJF con prelazione:

- ◆ $P_1(0:2), P_2(2:8) P_3(8:12), P_4(12:16), P_5(16:22), P_1(22:30)$
- ◆ $T = [30+(8-2)+(12-6)+(16-7)+(22-8)]/5 = 13$

(c) FCFS:

- ◆ $P1(0:10) P2(10:16) P3(16:20) P4(20:24) P5(24:30)$
- ◆ $T = [10+(16-2)+(20-6)+(24-7)+(30-8)]/5 = 18,6$

(d) Priorità:

- ◆ $P1(0 :10) P4(10 :14) P3(14:18) P5(18:24) P2(24:30)$
- ◆ $T = [10+(30-2)+(18-6)+(14-7)+(24-8)]/5 = 14,6$

Esercizio 2

- Supponiamo che il sistema di paginazione utilizzato dal sistema operativo assegni un numero fissato di frame (blocchi) di dimensione 512B a ciascun processo e che l'algoritmo di sostituzione delle pagine su richiesta sia di tipo LRU. Prendiamo in considerazione il seguente programma:

```
#define N 512
int a[N], c[N];
int i;

...
for (i=0; i < N/2; i++)
    a[2 * i] = c [N/2 + i] + c[N/2-i-1];
...
```

- Si assuma che:
 - ◆ un intero sia 4B, il codice e le dichiarazioni di i ed N siano collocate in pagine tali che nessun accesso a tali entità provochi un page-fault.
 - ◆ il programma abbia bisogno, esclusi i dati, di 12KB di memoria, di cui 1KB di stack.
 - ◆ ci sono 3 eseguibili diversi: con N=512, 1024 e 2048, che hanno bisogno, esclusi i dati, di 12KB di memoria (di cui 1KB di stack) e che vengono mandati in esecuzione uno dietro l'altro.

Esercizio 2

- Rispondere ai seguenti quesiti:
 - a) quanti page fault si verificano con 4 frame assegnati per i dati?
 - b) quanti frame sono necessari per ottenere il minimo di page fault?
 - c) qual'è la dimensione minima del TLB affinché l'hit ratio sia il massimo?
 - d) se sono disponibili 60 frame da 1KB ciascuna, nel caso di allocazione proporzionale, quanti frame ottiene ciascun processo?

Soluzione

La mappa della memoria é

A(0:127)|A(128:255)|A(256:383)|A(384:511)|C(0:127)|C(128:255)|C(256:383)|C(384:511)|

La stringa dei riferimenti è

6 5 0 6 5 1 7 4 2 7 4 3

la stringa delle distanze è

∞ ∞ ∞ 3 3 ∞ ∞ ∞ ∞ 3 3 ∞

a. Tanti quanti sono i riferimenti: $C_4 = 12$.

b. $m = 3$ corrispondente a $F_3 = 8$

c. Il numero di elementi nel TLB è quindi 3

d. $s_1 = 12 + 2 = 14K$ $s_2 = 12 + 4 = 16K$ $s_3 = 12 + 8 = 20K$

Quindi: $S = 14 + 16 + 20 = 50$ e

$a_1 = (14/50)*60 = 16$ $a_2 = (16/50)*60 = 19$ $a_3 = (20/50)*60 = 24$

Esercizio 3

- Si scriva uno pseudo-codice che implementi la seguente situazione: un processo padre genera un figlio e attende la sua terminazione; il figlio a sua volta genera un nipote e termina; il nipote attende 10 secondi e a sua volta termina.

Si indichi e commenti a quali cambiamenti di stato vanno incontro i processi coinvolti.

Soluzione

```
#include <unistd.h>
#include <stdio.h>
int nipote, stato;

main (void) {
int figlio;
switch (figlio=fork()) {
    case -1: /* errore fork 1 */
        perror("fork 1");
        exit(1);
    case 0: /* figlio */
        switch (nipote=fork()) {
            case -1: /* errore fork 2 */
                perror("fork 2");
                exit(2);
            case 0: /* nipote */
                sleep(10);
                break;
            default: /* figlio */
                exit(0);
        }
        break;
    default: /* padre */
        wait(&stato);
        exit(0);
}}
}
```

Soluzione

- Il padre è running e fa fork, quindi va in stato di wait fino a che il figlio ritorna; al termine, ritorna nello stato ready da cui viene scelto per andare in running, quindi termina. Concorrentemente il figlio è ready, quindi running, quindi terminated; Concorrentemente il figlio è ready, quindi running, quindi terminated;;
- La differenza tra figlio e nipote è che il nipote, morto il figlio, viene ereditato da init e quindi non passa nello stato zombie.