Esame di Sistemi Operativi
AA 2017/18
19 Giugno 2018

Nome	Cognome	Matricola

Istruzioni

Scrivere il proprio nome e cognome su ogni foglio dell'elaborato. Usare questo testo come bella copia per le risposte, utilizzando l'apposito spazio in calce alla descrizione dell'esercizio.

Esercizio 1

Sia data la seguente tabella che descrive il comportamento di un insieme di processi

processo	**tempo di inizio**	**CPU burst 1**	**IO burst 1**	**CPU burst 2**	**IO burst 2**
P1	0	5	5	3	1
P2	1	2	5	2	5
P3	5	8	1	8	1
P4	7	1	9	1	9

Domanda Si assuma di disporre di uno scheduler preemptive con quanto di tempo 5, e politica di selezione dei processi *Shortest Remaining Job First* (SRJF). Si assuma che i processi in entrata alla CPU "dichiarino" il numero di quanti necessari all' esecuzione di un CPU burst. Si assuma inoltre che:

- l'operazione di avvio di un processo lo porti nella coda di ready, ma **non** necessariamente in esecuzione
- il termine di un I/O porti il processo che termina nella coda di ready, ma **non** in esecuzione.

Si illustri il comportamento dello scheduler in questione nel periodo indicato, avvalendosi degli schemi di seguito riportati (vedi pagina seguente).

Soluzione Date queste premesse, la traccia di esecuzione dei processi e' riportata nella Figura 1. Si e' assunto che nel caso vi siano piu' processi con caratteristiche uguali vada in esecuzione quello con identificativo piu' basso.

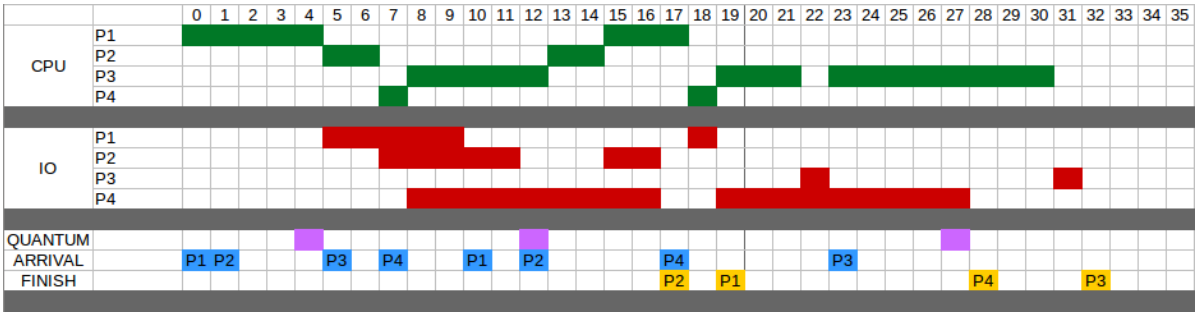


Figure 1: Traccia di esecuzione dei processi schedulati tramite *quantized SRJF*. In verde sono indicati i cicli di CPU burst e in rosso quelli di I/O; il time quantum di 5 cicli e' scandito in colore viola, l'arrivo dei processi in azzurro e la conclusione di un processo in giallo.

Nome	Cognome	Matricola

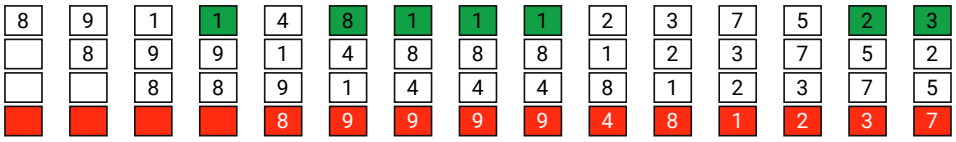
Esercizio 2

Sia data la seguente traccia di accesso alle pagine di memoria:

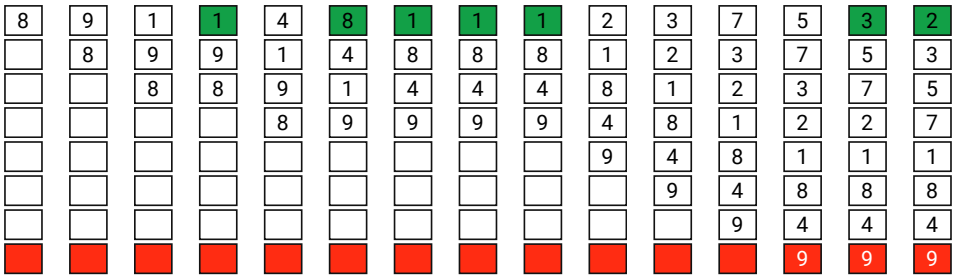
8 9 1 1 4 8 1 1 1 2 3 7 5 2 3.

Domanda Si assuma di avere un TLB di 4 slot gestito in modalita' LRU. Di quanto migliorano le prestazioni della memoria raddoppiandolo?

Soluzione La tracce di accesso nei due casi sono raffigurate rispettivamente nelle Figure 2a e 2b.



(a) Traccia con TLB a 4 slot



(b) Traccia con TLB a 8 slot

Figure 2: Illustrazione della traccia di accesso alle pagine di memoria. In verde indichiamo un *hit* nel TLB, mentre in rosso viene indicata la pagina candidata ad eliminizione - secondo la politica LRU.

Quindi, supponendo che il TLB abbia 4 unita', il tempo di accesso medio T^4_{mean} e' dato dalla seguente relazione:

$$T^4_{mean} = 7T_{hit} + 8Tmiss$$

dove

$$T_{hit} = T_{TLB} + T_{fetch}$$
$$T_{miss} = 2T_{TLB} + 2T_{fetch}$$

Analogamente, nel caso in cui il TLB abbia 8 unita', il tempo medio sara' calcolato come segue:

$$T^8_{mean} = 7T_{hit} + 8Tmiss$$

E' facilmente intuibile da un semplice confronto che in questo particolare caso aumentare la dimensione del TLB non influenza le prestazioni.

Nome	Cognome	Matricola

Esercizio 3

Cosa stampa il seguente programma?

```c
#define N 3
int v[N];

int main () {
    for (int i=0; i<N; ++i)
        v[i]=i;

    for (int i=0; i<N; ++i) {
        int child_pid=fork();
        if (! child_pid) {
            for (int j=0; j<N; ++j) {
                v[j]=i;
            }
            exit(0);
        }
        int retval;
        wait(&retval);
    }

    printf("[");
    for (int i=0; i<N; ++i)
    printf("%d ", v[i]);
    printf("]\n");
}
```

Soluzione La `exit` posta alla fine del ciclo interno impedisce di effettuare le stampe ai processi figli. A causa di cio', l'output del programma è il seguente:

```
[0 1 2]
```

Nome	Cognome	Matricola

Esercizio 4

Sia dato un sottosistema di memoria con paginazione, caratterizzato dalle seguenti dimensioni:

- frame 4KB
- memoria fisica indirizzabile 64GB

Domande

- Si calcoli il numero di bit necessari per individuare una pagina in un indirizzo virtuale
- Considerato che il tempo di accesso medio ad una pagina e' 120 ns, un accesso al TLB impiega 1 ns, e la probabilita' di page fault e' 1e-3, si calcoli il tempo di un ciclo di lettura/scrittura.

Soluzione

- La dimensione di un frame e' pari a 4KB, per cui necessita di 12 bit ($2^{12} = 4K$); il numero di bit necessari ad indirizzare 64GB di memoria fisica invece sono 36 ($2^{36} = 64G$). Percio', in tale sistema, il numero di bit per il page number sara' $36 - 12 = 24$.
- La formula generica del *Effective Access Time* e' data dalla relazione

$$EAT = p(T_{TLB} + T_{fetch}) + (1 - p)(2T_{TLB} + 2T_{fetch}) \tag{1}$$

  dove p indica l'hit ratio del sistema. Dalla (1) potremo calcolare T_{fetch} sostituendo i valori numerici ai parametri, ottenendo il seguente valore:

$$T_{fetch} = \frac{119001}{1001} \approx 118.88\ ns$$

Nome	Cognome	Matricola

Esercizio 5

Descrivere con un breve esempio un File System con allocazione a indice.

Soluzione Un File System ha tra i suoi scopi l'allocazione e la gestione dello spazio sul disco per tutti i file in esso contenuti. Nel caso di *Indexed Allocation* i blocchi di un file saranno posti in maniera scattered sul disco (e non contigua), analogamente a quanto avviene nella allocazione a lista (*Linked Allocation*). In questo caso pero' ogni file conterra' un *index block*, ovvero un blocco contenente i puntatori a tutti gli altri blocchi componenti il file.

Quando un file viene creato, tutti i puntatori dell'index block sono settati a `null`; quando un nuovo blocco viene richiesto e scritto, il puntatore a tale blocco entrera' nell'index block.

Tale tipo di allocazione permette di guardagnare velocita' rispetto ad una implementazione tramite linked list nel caso in cui si effettuino molti accessi scattered ai blocchi (non bisogna scorrersi tutta la lista, ma basta fare una ricerca). Il costo da pagare e' lo spazio necessario a contenere l'index block stesso.

Un esempio di tale implementazione e' riportato nella Figura 3.

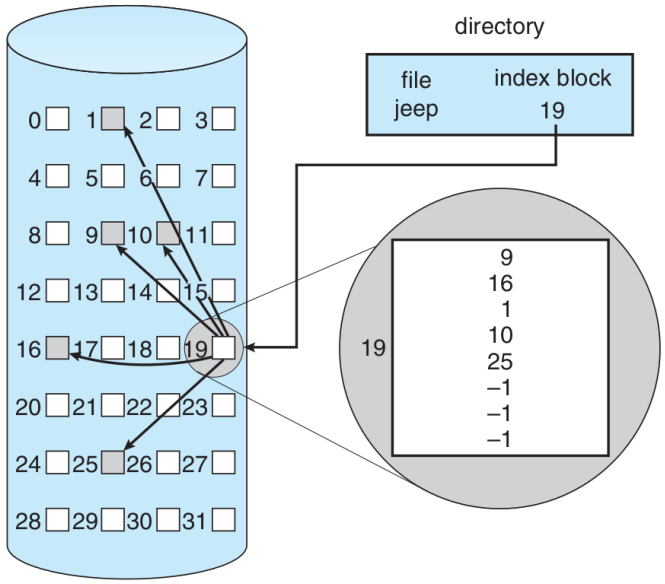


Figure 3: Esempio di allocazione indicizzata: nel primo blocco del file `jeep` si trova l'index block contente i puntatori a tutti gli altri blocchi del file.

Nome	Cognome	Matricola

Esercizio 6

Sia data la seguente tabella di processi/risorse.

Allocated:

Processo	R1	R2	R3	R4
P0	1	6	0	3
P1	1	6	0	1
P2	3	0	2	0

Max:

Processo	R1	R2	R3	R4
P0	7	12	3	5
P1	3	12	2	3
P2	9	0	2	0

Available:

R1	R2	R3	R4
7	1	1	5

Domande Si illustri l'evoluzione dell'algoritmo del banchiere nella gestione dei deadlock.

Soluzione *Non esiste una soluzione safe* ottenibile tramite l'algoritmo del banchiere. Di seguito l'evoluzione dell'algortimo:

```
*****************************************
op: 0
next_op: 1
p: 0
work:
SIZE: 4
7 1 1 5
needed:
SIZE: 4
6 6 3 2
reject

*****************************************
op: 0
next_op: 2
p: 1
work:
SIZE: 4
7 1 1 5
needed:
SIZE: 4
2 6 2 2
reject

*****************************************
op: 0
next_op: 3
p: 2
work:
SIZE: 4
```

```
7 1 1 5
needed:
SIZE: 4
6 0 0 0
accept

****************************************
op: 1
next_op: 2
p: 0
work:
SIZE: 4
10 1 3 5
needed:
SIZE: 4
6 6 3 2
reject

****************************************
op: 1
next_op: 3
p: 1
work:
SIZE: 4
10 1 3 5
needed:
SIZE: 4
2 6 2 2
reject
unsafe

is_safe: 0
```

Nome	Cognome	Matricola

Esercizio 7

Cos'e' un *Process Control Block* (PCB)? Cosa contiene al suo interno?

Soluzione Il PCB e' una struttura dati contente tutte le informazioni relative al processo a cui e' associato. Esempi di informazioni contenute nel PCB sono:

- stato del processo (running, waiting, zombie ...)
- Program Counter (PC), ovvero il registro contente la prossima istruzione da eseguire
- registri della CPU
- informazioni sulla memoria allocata al processo
- informazioni sull'I/O relativo al processo.

Il PCB e' fondamentale durante il context switch, permettendo di salvare "lo stato" del processo (PC, registri ecc.) e di ripristinare l'esecuzione da dove la si era interrotta.

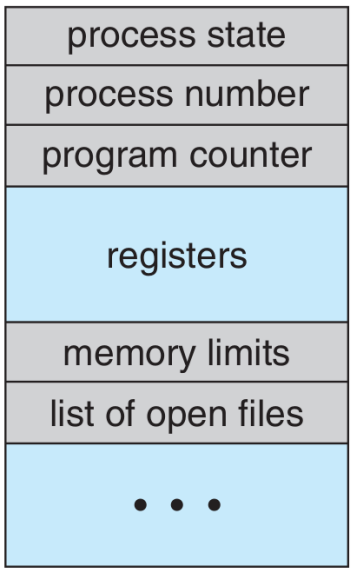


Figure 4: Una rappresentazione grafica del PCB. Esso contiene tutte le informazioni relative alla gestione di un processo da parte del sistema operativo.

Nome	Cognome	Matricola

Esercizio 8

Cos'e' una coda di messaggi IPC? Fornire un breve esempio del suo utilizzo.

Soluzione Tra i metodi di IPC (Inter-Process Communication) troviamo lo scambio di messaggi tra piu' processi. Rispetto all'implementazione tramite shared memory, si avranno numerosi vantaggi tra i quali:

- maggiore flassibilita' - e.g. e' possibile implementare una comunicazione remota tra processi eseguiti su macchine diverse
- semplicita' di implementazione
- sicurezza nella comunicazione - e.g. nessuna lettura parziale dei dati poiche' processati messaggio per messaggio

Il costo da pagare, invece, sara' in termini di performances:

- maggiore overhead poiche' la comunicazione dovra' essere gestita dal sistema operativo
- copia dei dati nella coda
- l'accesso ai dati e' in generale piu' lento perche' limitato ad un messaggio per volta (la dimensione di un messaggio e' generalmente limitata e relativamente piccola).

La comunicazione e' basata sulle operazioni di `send` e `receive`. Tali funzioni (syscall) possono essere sviluppati in base alle specifiche esigenze dell'utente (comunicazione sincrona/asincrona, diretta/indiretta, limitata/illimitata ...).

Alla luce di quanto detto finora, una coda di messaggi e' un oggetto gestito dal sistema operativo che implementa una *mailbox*. I processi quindi potranno creare una coda di messaggi oppure linkarsi ad una mailbox esistente a partire da un identificatore della coda. Le altre operazioni fondamentali sono:

- ◇ check dello status della coda
- ◇ post di un messaggio
- ◇ attesa di un messaggio (bloccante o non bloccante).

Nome	Cognome	Matricola

Esercizio 9

Si consideri un Sistema Operativo batch, avente una tabella di processi di dimensione 20. Si assuma che i job durino in media 10s.

Domanda Ogni quanto tempo il sistema puo' accettare un nuovo job senza eccedere il numero di PCB disponibili?

Soluzione La Legge di Little permette di mettere in relazione dimensione della coda dei processi, tempo medio di esecuzione e frequenza media di arrivo degli stessi; tale formula e' descritta dalla relazione:

$$n = \lambda \cdot W \tag{2}$$

dove n indica la dimensione della coda, λ la frequenza di arrivo media e W il tempo di esecuzione medio. La soluzione e' facilmente ottenibile per sostituzione dalla (2), ottenendo una frequenza media pari a $\lambda = 2Hz$ - ovvero 2 processi al secondo.

Nome	Cognome	Matricola

Esercizio 10

Cos'e' la tabella delle syscall? Cos'e' l'interrupt vector?

Soluzione L'*interrupt vector* e' un vettore di puntatori a funzioni; queste ultime saranno le *Interrupt Service Routine* (ISR) che gestiranno i vari interrupt.

Analogamente, la tabella delle syscall conterra' in ogni locazione il puntatore a funzione che gestisce quella determinata syscall. Alla tabella verra' associata anche un vettore contenente il numero e l'ordine di parametri che detta syscall richiede. Per registrare una nuova syscall, essa va registrata nel sistema ed aggiunta al vettore delle syscall del sistema operativo, specificando il numero di argomenti (ed il loro ordine) nell'altro vettore di sistema apposito.