Esame 2021/07/05

Blascovich Alessio

Domanda 1

MAX PUNTI: 6
Domanda:

Si descriva il concetto di paginazione.
In seguito data la stringa 0, 1, 2, 3, 0, 2, 1, 2, 1, 0, 1, 2, 3 calcolare il numero di page fault con gli algoritmi FIFO, LRU con 3 frame a disposizione, mostrare il contenuto della memoria passo passo.
Soluzione:

La paginazione è una tecnica per la gestione della memoria che punta ad eliminare la frammentazione esterna.
Questo è reso possibile grazie alla non continuità dello spazio assegnato ad un processo, così facendo si può allocare qualsiasi frame di memoria a patto che sia libero.
Divide la memoria fisica in blocchi di dimensione fissa detti *frame* e la memoria logina in blocchi di uguali dimensioni detti *pagine*.
Per avviare un processo grande n pagine devo trovare n frame liberi, per effettuare questa ricerca si usa una cosidetta *page table* cioè una tabella che tiene traccia a quale processo è associato ogni frame.
Si assuma pf l'acronimo di page fault.

- *FIFO*:
  Il metodo *FIFO* gestisce la memeoria come una semplice coda, il primo segmaneto ad essere entrato è il primo ad essere fatto uscire.

0	null	null

$\xrightarrow{pf=1}$

0	1	null

$\xrightarrow{pf=2}$

0	1	2

$\xrightarrow{pf=3}$

3	1	2

$\xrightarrow{pf=4}$

3	0	2

$\xrightarrow{pf=5}$

3	0	1

$\xrightarrow{pf=6}$

2	0	1

$\xrightarrow{pf=7}$

2	0	1

$\xrightarrow{pf=7}$

2	0	1

$\xrightarrow{pf=7}$

2	0	1

$\xrightarrow{pf=7}$

2	0	1

$\xrightarrow{pf=7}$

2	3	1

$\xrightarrow{pf=8}$ Termino con un totale di 8 page fault.

- *LRU*:
  L'algoritmo *LRU* tiene associato ad ogni frame il clock della CPU nell'istante in cui quel frame è stato usato per l'ultima volta, verrà tolto il frame che non viene usato da più tempo.
  Nella prima riga sono indicati gli istanti, mentre nella seconda il numero del frame.

1	null	null
0	null	null

$\xrightarrow{pf=1}$

1	2	null
0	1	null

$\xrightarrow{pf=2}$

1	2	3
0	1	2

$\xrightarrow{pf=3}$

4	2	3
3	1	2

$\xrightarrow{pf=4}$

4	5	3
3	0	2

$\xrightarrow{pf=5}$

4	5	6
3	0	2

$\xrightarrow{pf=5}$

7	5	6
1	0	2

$\xrightarrow{pf=6}$

7	5	8
1	0	2

$\xrightarrow{pf=6}$

9	5	8
1	0	2

$\xrightarrow{pf=6}$

9	10	8
1	0	2

$\xrightarrow{pf=6}$

11	10	8
1	0	2

$\xrightarrow{pf=6}$

11	10	12
1	0	2

$\xrightarrow{pf=6}$

11	13	12
1	3	2

$\xrightarrow{pf=7}$ Termino con un totale di 7 page fault.

Domanda 2

MAX PUNTI: 6
Domanda:

Spiegare nel dettaglio che casi di frammentazione esterna/interna posso avere con la paginazione.
Spiegare perchè il fyle system di tipo FAT non è efficente per l'accesso diretto e per file di grandi dimensioni.
Soluzione:

Onestamente non ho capito la prima parte della domanda, perchè la paginazione in pratica ha solo un minimo di frammentazione interna e 0 frammentazione esterna.
FAT è un tipo di file system usato in passato da microsoft, si basa su una lista concatenata ma che anzichè puntare ad un singolo blocco punta ad un'area di disco contigua.
Questo migliora l'accesso casuale ma l'efficenza rimane comunque paragonabile a quella dell'allocazione a lista.
La dimensione massima di un file in FAT32 è 4GB questo per motivi di efficenza(tempi di risposta), infatti è pur sempre

una lista concatenata quindi devo scorrere tutto il file per arrivare alla fine di esso.
Infatti se mi devo spostare all'interno di uno stesso insieme di blocchi(extent) allora il tempo di acesso è costante mentre se devo cercare un blocco al di fuori di un extent il tempo diventa lieare.

Domanda 3

MAX PUNTI: 6
Domanda:

Dato un calcolatore con memoria virtuale che sfrutta la segmentazione paginata con pagine da 8KB, 7 segmenti(1023 pagine ogni frammento) e una memoria fisica da 512MB con 32bit per l'indirizzamento.
Si calcoli:

1. Quanti bit devo usare per rappresentare gli indirizzi virtuali?
2. Quanti bit devo usare per rappresentare gli indirizzi fisici?

Soluzione:

1. So che una pagina è grande 8KB=8192B=2^{13} $\Rightarrow$ devo dare 13 bit per l'offset.
   Ho 7 segmenti quindi $7 \leq 2^3$ $\Rightarrow$ devo assegnare 3 bit per indicare il frammento.
   Dopodichè devo sapere in che pagina mi trovo, sapendo che ho 1023 pagine per frame $1023 \leq 2^{10}$ quindi mi servono 10 bit per sapere in che pagina sono.
   Ho quindi un totale di 13+3+10=26 bit per rappresentare un indirizzo virtuale.
2. 512MB=536870912B=2^{29} $\Rightarrow$ devo usare 29 bit per rappresentare la memoria fisica.

Domanda 4

MAX PUNTI: 6
Domanda:

Definire il deadlock e spiegare quali sono le condizioni sufficienti e necessarie perchè si verifichi.
Soluzione:

Un insieme di processi viene detto in deadlock solo quando tutti i processi di quel gruppo sono in attesa di un evento che può essere generato da un processo di quel gruppo.
Perchè si verifichi un deadlock devono essere vere tutte e 4 le casistiche seguenti:

1. *Mutua esclusione*: una o più risorse devono essere non condivisibili tra i processi.
2. *Hold-and-wait*: un processo `P1` deve attendere che un secondo processo `P2` liberi una risorsa mentre lui ne detiene un'altra.
3. *No preemption*: le risorse possono essere rilasciate solo volontariamente dai processi.
4. *Attesa circolare*: deve esistere un insieme di processi che attende il liberarsi di una risorsa in modo ciclico.

Domanda 4

MAX PUNTI: 8
Domanda:

Scrivere in pseudocodice un algoritmo basato sull'uso di semafori che sincronizzi i processi `P`, `C1` e `C2`.
I processi sono organizzati secondo lo schema produttore/consumatore, `P` scrive sul buffer, mentre `P1` e `P2` leggono.
Il processo `C1` ha sempre la precedenza sul buffer, quindi se `C2` prova ad accedervi ma se anche `C1` ci sta provando allora `C1` ha la precedenza.
Infine su supponga che i processi siano in funzione perpetua `while(true){...}`.
Soluzione:

Per questo probelma è possibile usare due semafori binari, uno per ogni processo consumatore.

```
main(){
    int N=256;
    char[] buffer=new char[N];
    Semaforo_int bufferPieno=N;
```

```
    Semaforo_int bufferVuoto=0;
    Semaforo_bin controlloBuffer=1;
    Seamforo_bin gestionePrecedenza=1;
    bool precedenza=false;
    while(true){
        P(){
            P(bufferPieno);
            P(controlloBuffer);
            //scrivi sul buffer
            V(controlloBuffer);
            V(bufferVuoto);
        }

        C1(){
            P(gestionePrecedenza);
            precedenza=true;
            V(gestionePrecedenza);

            P(controlloBuffer);
            P(bufferVuoto);
            //lettura buffer
            V(bufferPieno);
            V(controlloBuffer);

            P(gestionePrecedenza);
            precedenza=false;
            V(gestionePrecedenza);
        }

        C2(){
            P(controlloBuffer);
            P(bufferVuoto);
            if(precedenza==true){
                V(controlloBuffer);
            }
            else{
                //lettura buffer
                V(controlloBuffer);
                V(bufferVuoto);
            }
        }
    }
}
```