Esame 2021/06/25

Blascovich Alessio

Domanda 1

MAX PUNTI: 6
Domanda:
Dare la definizione di binding degli indirizzi e spiegare le tipologie di collegamento(linking) e caricamento(loading).
Soluzione:
Per binding degli indirizzi si definisce la traduzione degli indirizzi logici usati da un problema a indirizzi fisici usati da quel programma una volta che viene trasformato in processo, il binding può essere fatto in diverse fasi:

- *Compile time*: richiede di conoscere precedentemente le locazioni di memeoria nelle quali verrà caricato il programma, se le locazioni di memoria variano allora bisogna ricompilare il programma.
- *Load time*: generazione degli indirzzi in base alla prima locazione di memoria occupata dal programma, di conseguenza gli indirizzi saranno della forma indirizzo_inizio_programma+offset.
  Però se il programma viene sspostato devo ricompilare il programma.
- *Run time*: il binding è posticipato per permettere al processo di essere spostato in memoria durante la sua esecuzione, questo metodo richiede però supporto hardware aggiuntivo.

Il *linking* può avvenire in due modi diversi:

1. *Statico*: durante la compilazione viene caricata un copia intera delle librerie usate, quindi tutti i riferimenti sono risolti durante la compilazione.
2. *Dinamico*: il codice in se contiene dei puntatori(stub) alla funzione esterna chiamata. Un esempio di questo metodo sono i file .ddl(*Dynamic Linked Library*) usate da Windows.

Il *loading* può essere effettuato in due modi diversi:

1. *Statico*: l'intero programma viene caricato per intero in memoria.
2. *Dinamico*: il processo viene diviso in sub routine così da caricarle in memoria solo quando necessario.

Domanda 2

MAX PUNTI: 6
Domanda:
Si consideri un sistema con 3 processi P_1, P_2 e P_3 con 3 tipi di risorse A, B e C.
Si supponga che all'istante T il sistema si trovi nella seguente condizione:

ALLOC

processo	A	B	C
P_1	2	2	3
P_2	2	0	3
P_3	1	2	4

MAX

processo	A	B	C
P_1	3	6	8
P_2	4	3	3
P_3	3	4	4

Si supponga che allo stesso istante T siano ancora disponibli 2 di A, 3 di B e 0 di C.

a Il sisetma è in stato safe?

b All'istante T il sistema può soddisfare la richiesta $P_1(1, 0, 1)$?

c All'istante T il sistema può soddisfare la richiesta $P_1(2, 0, 0)$?

Soluzione:

a Per verificare se il sistema è in uno stato safe uso l'algortimo del banchiere senza la parte che considera una possibile futura richiesta. L'algortimo del banchiere crea da prima una tabella *Nedd* definita come $Nedd[]=MAX[]\text{-}ALLOC[]$. Devo inoltre porre il vettore $work[]=(2,3,0)$.

Need

processo	*A*	*B*	*C*
P_1	1	4	5
P_2	2	3	0
P_3	2	2	0

Adesso devo verificare di essere in uno stato safe con i seguenti passaggi:

1. Scorro gli elementi di *Need* fino a trovare una righa tale che $Need[] \leq Work[]$.
2. Trovata la riga eseguo $Work[]=Work[]+ALLOC[P_i]$, torno al punto 1 escludendo dalle prossime valutazioni la riga nella quale mi sono precedentemente fermato.
3. Se arrivo fino alla fine senza aver trovato una riga $Need[] \leq Work[]$ allora **non** sono in uno stato safe.

Inizio a iterare l'algoritmo.

- prima iterazione
  (1,4,5) $\leq$ (2,3,0) false $\Rightarrow$ ++i
  (2,3,0) $\leq$ (2,3,0) true $\Rightarrow$ $Work=Work+ALLOC[P_B]=(4,3,3)$.
- seconda iterazione
  (1,4,5) $\leq$ (4,3,3) false $\Rightarrow$ ++i
  (2,2,0) $\leq$ (4,3,3) true $\Rightarrow$ $Work=Work+ALLOC[P_C]=(5,5,7)$.
- terza iterazione
  (1,4,5) $\leq$ (5,5,7) true $\Rightarrow$ $Work=Work+ALLOC[P_A]=(7,7,10)$.

Ho concluso tutte le iterazioni servendo per ogni iterazione un processo, quindi sono in uno stato safe.

b Ora simulo la richiesta $P_1(1,0,1)=Req$.
Devo verificare che $\forall i$ vale che $Req[] \leq Need[i][]$ e $Req[] \leq Work[]$.

- $Req[] \leq Need[1][] \Rightarrow$ (1,0,1)$\leq$(1,4,5) true.
- $Req[] \leq Need[2][] \Rightarrow$ (1,0,1)$\leq$(2,3,0) false.

Essendo un requisito non soddisfatta mi fermo subito.

c Ora simulo la richiesta $P_1(2,0,0)=Req$.
Devo verificare che $\forall i$ vale che $Req[] \leq Need[i][]$ e $Req[] \leq Work[]$.

- $Req[] \leq Need[1][] \Rightarrow$ (2,0,0)$\leq$(1,4,5) false.

Essendo un requisito non soddisfatta mi fermo subito.

Domanda 3

MAX PUNTI: 6
Domanda:
Descrivere nel dettaglio il funzionamento del RAID di livello 5.
Soluzione:
Il RAID di 5° livello gestisce i dischi a livello di blocchi, i dati di ogni disco sono distribuiti tra tutti i dischi, e così via. Ogni disco al suo interno ha il blocco che contiene le parità di un altro disco del RAID.
Infatti al disco A è proibito avere al suo interno il blocco con le proprie parità.
E' simile al RAID di 4° livello solo che risolve il problema del singolo disco che contiene tutti i bit di parità, però la scrittura sui singoli dischi rimane comunque molto lenta come nel RAID di 4° livello.

Domanda 4

MAX PUNTI: 6
Domanda:

Descrivere le principali differenze tra un processo ed un thread, facendo l'esempio di un caso pratico in cui conviene usare un thread anzichè un processo.
Soluzione: Un processo è associato con uno spazio di indirizzi e a delle risorse del sistema, per questo quando io creo un processo figlio implicitamente creo una copia di tutte le variabili del processo padre e alloco delle nuove risorse al figlio.
Un thread è associato ai valori dei registri, al program counter(PC) al suo stato di esecuzione e allo stato dello stack.
Infatti un thread condivide lo spazio degli indirizzi con il processo da cui è stato generato e anche le risorse allocate al processo padre.
Un thread ha molti vantaggi:

- Riducono il tempo di risposta perchè il thread non è bloccante.
- Ha una comunicazione aggevolata col padre perchè condivide con esso lo spazio degli indirizzi.
- La Creazione/distruzione/switch di thread è molto più veloce che tra singoli processi.
- I thread sono molto scalabili, infatti aumentano il livello di parallelismo dei processi se la macchina è multiprocessore.

A livello pratico multissimi programmi usano i thread per esempio grazie ai thread io posso navigare su internet mentre ascolto il contenuto di un file audio o scrivere un file di testo mentre ho aperto un riproduttore di video.

Domanda 5

MAX PUNTI: 8
Domanda:

Dato un calcolatore con due gruppi di processi ed un buffer il quale è condiviso tra i due gruppi.
Un insieme di processi detto `Reader` è consentito accedere al buffer solo in modo mutualmente esclusivo con altri processi dello stesso gruppo e con processi del gruppo `Writer`.
Scrivere uno pseudocodice che permette l'accesso al buffer dando precedenza ai processi del gruppo `Reader`, quindi un `Reader` deve attendere solo e solo se un `Writer` è già in esecuzione.
Finchè arrivano `Reader` i `Writer` devono attendere.
Soluzione:

```
Semaforo_bin usoBuffer=1;
Semaforo_bin usoPrecedenza=1;
bool precedenza=false;

while(true){
   Reader(){
      P(usoPrecedenza);
      precedenza=true;
      V(usoPrecedenza);

      P(usoBuffer);
      //modifica buffer
      V(usoBuffer);

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

   Writer(){
      P(usoPrecedenza);
      if(precedenza==true){
         V(usoPrecedenza);
      }
      else{
```

```
            P(usoBuffer);
            //modifica buffer
            V(usoBuffer)
        }
    }
}
```