Sistemi Operativi
22 marzo 2007
Compitino 1/A

Si risponda ai seguenti quesiti, giustificando le risposte.

1. Si descrivano le caratteristiche della computazione nei seguenti sistemi operativi:
   (a) batch;
   (b) multiprogrammati;
   (c) time-sharing;
   (d) distribuiti.

   Risposta:
   Batch. I job sono eseguiti uno alla volta; i programmi di un job sono eseguiti in modo sequenziale.

   Multiprogrammati. I programmi sono eseguiti in interleaving, la CPU commuta tra i vari programmi in memoria centrale, al fine di ottenere il massimo utilizzo della CPU.

   Time-sharing. I programmi dei vari utenti sono eseguiti in interleaving, gli utenti comunicano con il sistema attraverso una tastiera online, l'esecuzione dei programmi deve garantire tempi di risposta adeguati agli utenti.

   Distribuiti. I processi relativi ad un programma possono essere eseguiti in nodi diversi del sistema, al fine di sfruttare le risorse dell'intero sistema.

2. Si consideri un'applicazione per la prenotatione di voli aerei, che usa un certo numero di terminali di agenzia connessi ad un computer centrale. Il computer memorizza i dati in maniera centralizzata e permette ai terminali delle agenzie di accedervi in tempo reale. Discutere a livello qualitativo come deve essere implementata l'applicazione. Conviene usare processi o thread? Si possono verificare race condition?

   Risposta: L'applicazione può essere strutturata in processi multipli; ciascun processo serve un terminale di agenzia. Questa struttura permette ad un agente di un terminale di inserire le sue richieste, mentre altri agenti accedono e aggiornano i dati relativi alle prenotazioni. L'accesso a tali dati andrà disciplinato, al fine di evitare race condition (e.g. uno stesso posto prenotato da due diverse agenzie). Tutti i processi eseguono lo stesso codice e lavorano sulla stessa struttura dati, perciò è più efficiente usare thread invece di processi.

3. Un programma è costituito da un singolo loop che viene eseguito 50 volte. Il loop contiene una computazione che dura 50 ms, seguita da operazioni di I/O per 200 ms. Questo programma è eseguito su un sistema time-sharing insieme con altri 9 programmi identici. Lo scheduling della CPU è RR e l'overhead per l'operazione di context switch è di 3 ms. Si assuma che un processo che lascia la CPU per fare I/O prima di finire il suo quanto, al termine dell'operazione di I/O, venga rimesso in fondo alla coda con quanto intero.

   Si calcoli:
   (a) per q=50 ms, il tempo di risposta per la prima iterazione e per l'ennesima iterazione dell'i-esimo processo in coda;
   (b) per q= 20ms, il tempo di risposta per la prima iterazione e per la seconda iterazione dell'i-esimo processo in coda.

   Risposta:
   (a) Il tempo di risposta per la prima iterazione dell'i-esimo processo è $53i - 50$ ms. Il tempo di risposta per l'ennesima iterazione dell'i-esimo processo è $(53 \times 10) \times (n-1) + 53 \times (i-1) + 3 = 530n - 580 + 53i$ ms.
   (b) Il tempo di risposta per la prima iterazione dell'i-esimo processo è $23i - 20$ ms. Il tempo di risposta per la seconda iterazione dell'i-esimo processo è $23 \times 10 \times 2 + 13 \times 10 + 10 \times (i-1) + 3i = 580 + 13i$ ms.

4. Si consideri la seguente situazione, dove P_0, P_1, P_2, P_3 sono quattro processi in esecuzione, C è la matrice delle risorse correntemente allocate, Max è la matrice del numero massimo di risorse

assegnabili ad ogni processo e A è il vettore delle risorse disponibili:

	C			Max		
	A	B	C	A	B	C
P_0	1	0	4	4	2	7
P_1	2	0	1	4	4	5
P_2	0	3	2	2	4	2
P_3	1	4	1	4	6	4

Available (A)		
A	B	C
2	1	x

(a) Calcolare la matrice R delle richieste.

(b) Determinare il minimo valore di x tale che il sistema si trovi in uno stato sicuro.

Risposta:

(a) Matrice delle richieste $R = Max - C$:

	R	
A	B	C
3	2	3
2	4	4
2	1	0
3	2	3

(b) Il minimo valore di x tale da rendere lo stato sicuro è 2. Infatti, se $x = 0$, R_2 è l'unica riga della matrice delle richieste tale che $R_2 \leq A = (2, 1, 0)$ e quindi si può eseguire P_2 fino alla sua terminazione. A questo punto A diventa $A + C_2 = (2, 1, 0) + (0, 3, 2) = (2, 4, 2)$ e quindi si verifica immediatamente che non esiste nessuna riga residua della matrice delle richieste minore od uguale al nuovo valore di A. Se $x = 1$, allora nuovamente R_2 è l'unica riga della matrice delle richieste tale che $R_2 \leq A = (2, 1, 1)$ e quindi si può eseguire P_2 fino alla sua terminazione. A questo punto A diventa $A + C_2 = (2, 1, 1) + (0, 3, 2) = (2, 4, 3)$ e quindi si verifica immediatamente che non esiste nessuna riga residua della matrice delle richieste minore od uguale al nuovo valore di A. Infine, se $x = 2$, allora esiste la sequenza di esecuzione sicura $\langle P_2, P_1, P_0, P_3 \rangle$. Infatti, dapprima R_2 è l'unica riga della matrice delle richieste tale che $R_2 \leq A = (2, 1, 1)$ e quindi si può eseguire P_2 fino alla sua terminazione. A questo punto A diventa $A + C_2 = (2, 1, 2) + (0, 3, 2) = (2, 4, 4)$ e si può eseguire P_1 (dato che $R_1 \leq A$) aggiornando A il cui valore diventa $A + C_1 = (2, 4, 4) + (2, 0, 1) = (4, 4, 5)$. Eseguendo P_0 (dato che $R_0 \leq A$), si ottiene il nuovo valore di A: $A + C_0 = (4, 4, 5) + (1, 0, 4) = (5, 4, 9)$. Alla fine si può eseguire P_3 (dato che $R_3 \leq A$) ed ottenere il valore finale di A: $A + C_3 = (5, 4, 9) + (1, 4, 1) = (6, 8, 10)$.

5. Cosa si intende per *race condition* (corsa critica)? Il verificarsi di corse critiche rappresenta un evento positivo o negativo per un sistema di calcolo?

   Risposta: Per *race condition* (corsa critica) si intende la dipendenza del risultato di una o più operazioni dall'interleaving dei processi in esecuzione. Si tratta di un evento negativo perché rende non predicibile il comportamento di un sistema di calcolo.

6. Cosa si intende per *busy wait* (attesa attiva)? Si citi un aspetto positivo ed uno negativo di questa tecnica.

   Risposta: Per *busy wait* (attesa attiva) si intende il controllo ciclico del valore di una variabile che segnala il verificarsi di un evento. Un aspetto positivo è la facilità di implementazione, mentre un aspetto negativo è il consumo "inutile" del tempo di CPU per controllare il valore della variabile.

7. Descrivere brevemente il costrutto *monitor*.

   Risposta: Un *monitor* è un costrutto di sincronizzazione per linguaggi ad alto livello. La definizione di un monitor comporta la specifica di variabili (inizializzate), di funzioni (che rappresentano l'unico modo per manipolare le variabili del monitor) e di *condition variable* che servono a segnalare (*signal*) ed a mettersi in attesa (*wait*) di determinati eventi. Soltanto un processo o thread in ogni dato istante può eseguire una funzione del monitor (e quindi manipolare le variabili del monitor).

Sistemi Operativi
22 marzo 2007
Compitino 1/B

Si risponda ai seguenti quesiti, giustificando le risposte.

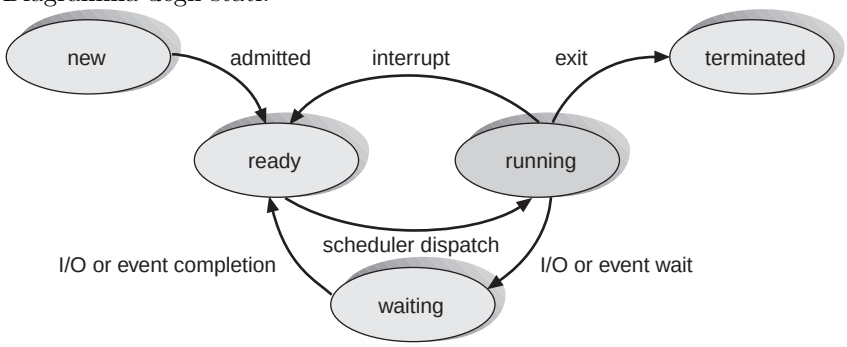
1. Che cos'è la tecnica dello spooling? Quali sono i vantaggi? Quando è stata introdotta?

   Risposta: Lo spooling (da Simultaneous Peripheral Operation On Line) è una tecnica per gestire autonomamente ed in modo efficiente molte richieste concorrenti a dispositivi di I/O che non possono accettare per loro natura flussi di dati intercalati. Ad esempio le applicazioni che vogliono stampare riversano i loro dati in un file che viene memorizzato in una directory di spool; più tardi un demone od un thread del kernel apposito provvederanno a leggere il contenuto dei file della directory e ad inviarlo in modo ordinato alla stampante.

   Quindi i vantaggi derivanti dall'adozione di questa tecnica consistono nel permettere di evitare ai processi lunghi tempi di attesa consentendo di coordinare input/output simultanei di dati. Lo spooling è stato introdotto per la prima volta ai tempi dei sistemi ad elaborazione a lotti (batch).

2. Si tracci il diagramma di transizione degli stati di un processo. Si elenchino le situazioni in cui una transizione di un processo può causare una transizione di un altro processo.

   Risposta: Diagramma degli stati:

   Situazioni in cui una transizione di un processo può causare una transizione di un altro processo:

   - In caso di scheduling della CPU con prelazione, un processo che passa da stato new in stato ready oppure da stato waiting a stato ready (per es. perché ha terminato un'operazione di I/O), può provocare il passaggio di un processo a minor priorità da stato running a stato ready.
   - Un processo che passa da stato running a stato terminated o a stato waiting (in attesa di un evento, per es. il completamento di un'operazione di I/O), provoca il passaggio di un processo da ready a running.

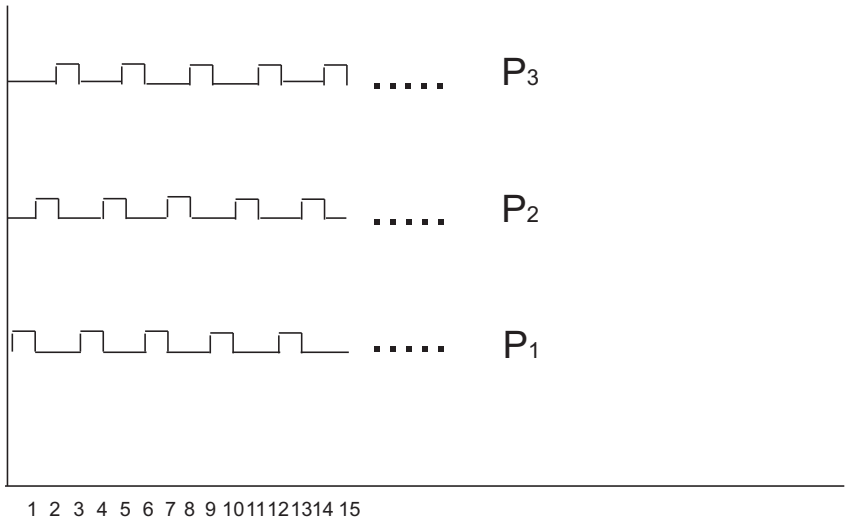
3. I processi P_1, P_2, P_3, con richieste di CPU totali di 10, 15, 20 s, sono in coda ready nell'ordine indicato. L'algoritmo di scheduling della CPU è con prelazione (e i processi prelazionati vengono messi in fondo alla coda ready). Ogni processo P_i ha un comportamento ciclico: in ogni ciclo consuma c_i secondi di CPU ed esegue operazioni di I/O per o_i secondi, dove $c_i = 2^{3-i}$ e $o_i = 0,5^{3-i}$. L'algoritmo di scheduling viene eseguito ad intervalli di 1 secondo. Si calcolino i tempi di turnaround per i processi, nel caso l'algoritmo di scheduling sia:

   (a) LCN (Least CPU Next), che seleziona il processo che ha usato meno tempo totale di CPU (a parità di priorità si seleziona in base all'ordine in coda);

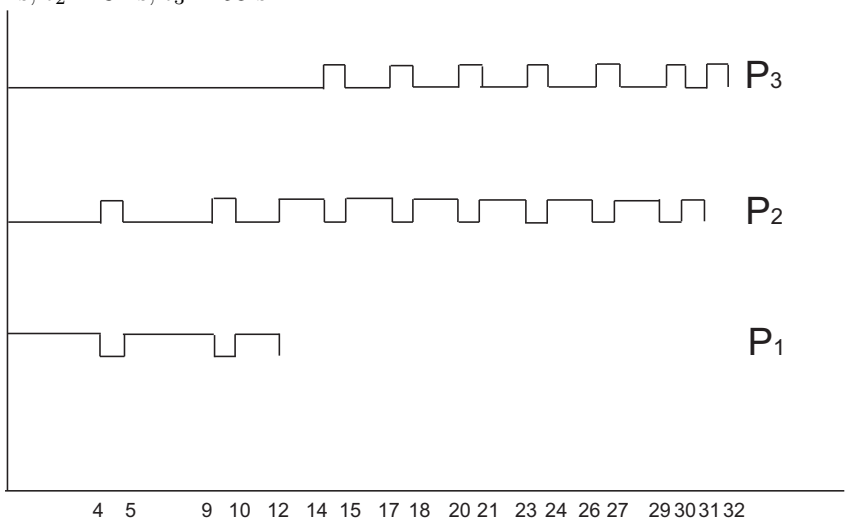
   (b) SRTF.

   Risposta:

   (a) $t_1 = 28$ s, $t_2 = 39$ s, $t_3 = 50$ s.



(b) $t_1 = 12$ s, $t_2 = 31$ s, $t_3 = 58$ s.



4. Si consideri la seguente situazione, dove P_0, P_1, P_2, P_3 sono quattro processi in esecuzione, C è la matrice delle risorse correntemente allocate, Max è la matrice del numero massimo di risorse assegnabili ad ogni processo e A è il vettore delle risorse disponibili:

	C			Max		
	A	B	C	A	B	C
P_0	1	0	4	4	2	7
P_1	2	0	1	4	4	5
P_2	0	3	2	2	4	2
P_3	1	4	1	4	6	4

Available (A)

A	B	C
3	x	1

(a) Calcolare la matrice R delle richieste.

(b) Determinare il minimo valore di x tale che il sistema si trovi in uno stato sicuro.

Risposta:

(a) Matrice delle richieste $R = Max - C$:

	R	
A	B	C
3	2	3
2	4	4
2	1	0
3	2	3

(b) Il minimo valore di x tale da rendere lo stato sicuro è 1. Infatti, se $x = 0$, nessuna riga della matrice delle richieste è minore o uguale al vettore A. Se $x = 1$, allora R_2 è l'unica riga della matrice delle richieste tale che $R_2 \leq A = (3, 1, 1)$ e quindi si può eseguire P_2 fino alla sua terminazione. A questo punto A diventa $A + C_2 = (3, 1, 1) + (0, 3, 2) = (3, 4, 3)$ e quindi si può eseguire P_0 fino alla sua terminazione. A questo punto A diventa $A + C_2 = (3, 4, 3) + (1, 0, 4) = (4, 4, 7)$ e si può eseguire P_1 (dato che $R_1 \leq A$) aggiornando A il cui valore diventa $A + C_1 = (4, 4, 7) + (2, 0, 1) = (6, 4, 8)$. Alla fine si può eseguire P_3 (dato che $R_3 \leq A$) ed ottenere il valore finale di A: $A + C_3 = (6, 4, 8) + (1, 4, 1) = (7, 8, 9)$.

5. Spiegare brevemente cosa si intende per *deadlock* (stallo) e cosa si intende per *starvation* (attesa indefinita).

   Risposta: Nonostante il problema della starvation sia strettamente correlato a quello del deadlock, si tratta di due cose differenti. Infatti, con il termine starvation, si indica un'assenza di progresso per cui un programma in esecuzione non riesce ad ottenere una risorsa od un servizio (a causa ad esempio della politica di allocazione della risorsa/servizio in questione adottata nel sistema), nonostante non venga mai bloccato. Invece con il termine deadlock si intende una situazione in cui un insieme di processi è bloccato in quanto ogni processo è in attesa di un evento che soltanto un altro processo dell'insieme può provocare. Dato che tutti i processi sono in attesa, nessuno di essi potrà mai produrre uno degli eventi che sbloccherebbero un altro elemento dell'insieme.

6. Cosa si intende per *inversione di priorità*?

   Risposta: Per *inversione di priorità* si intende una situazione in cui un processo a bassa priorità detiene una risorsa indispensabile ad un processo ad alta priorità. Di conseguenza si crea una situazione in cui un processo ad alta priorità deve attendere un processo a bassa priorità per proseguire.

7. Descrivere brevemente il funzionamento dei semafori e delle operazioni *up* (V) e *down* (P).

   Risposta: Un semaforo è una variabile che può assumere valori (sem) interi positivi o nulli e che può essere manipolata soltanto tramite due operazioni: *up* (V) e *down* (P).

   P(sem) (o down(sem)):

```
if(sem!=0)
  sem=sem-1;
else
  /* attendi finche' sem!=0 e decrementa sem di un'unita' */
```

   V(sem) (o up(sem)):

```
sem=sem+1; /* fai ripartire il primo fra i processi in coda
d'attesa */
```

   In questo modo un semaforo può essere utilizzato come costrutto per la sincronizzazione dei processi/thread.

Sistemi Operativi
22 marzo 2007
Compitino 2/A

Si risponda ai seguenti quesiti, giustificando le risposte.

1. Cosa si intende per anomalia di Belady? Si citi un algoritmo di rimpiazzamento delle pagine che soffre di questo problema ed uno che invece ne è immune.

   Risposta: Per anomalia di Belady si intende il fenomeno per cui, nonostante si incrementi la memoria fisica disponibile e quindi il numero di frame totali, non è detto che i page fault diminuiscano. Un algoritmo di rimpiazzamento delle pagine che soffre di questo problema è l'algoritmo FIFO (First-In First-Out), mentre LRU (Least Recently Used) e tutti gli algoritmi di stack ne sono immuni.

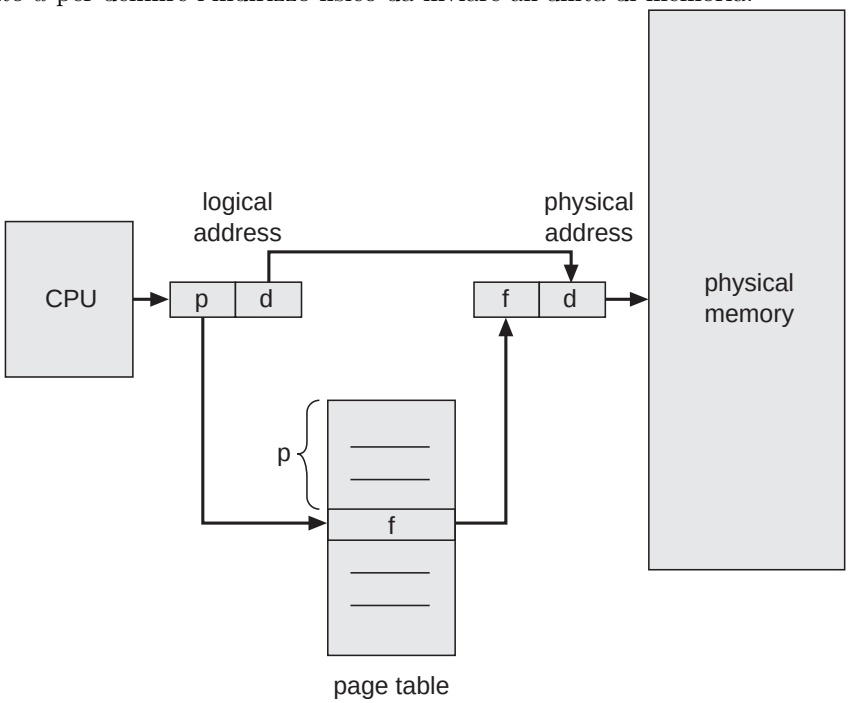
2. Si illustri brevemente il meccanismo di paginazione della memoria con il relativo schema di traduzione dell'indirizzo virtuale in indirizzo fisico.

   Risposta: La paginazione è una tecnica di allocazione della memoria non contigua che prevede:

   - la suddivisione della memoria fisica in *frame* (blocchi di dimensione fissa, una potenza di 2, tra 512 e 8192 byte);
   - la suddivisione della memoria logica in *pagine* (della stessa dimensione dei frame).

   Quindi, per eseguire un programma di n pagine, servono n frame liberi in cui caricare il programma. È compito del sistema operativo tenere traccia dei frame liberi. Non esiste frammentazione esterna e la frammentazione interna è ridotta all'ultima pagina allocata al processo.

   Per quanto riguarda la traduzione degli indirizzi logici in indirizzi fisici, ogni indirizzo generato dalla CPU si suddivide in un numero di pagina p e uno spiazzamento (offset) d all'interno della pagina. Il numero di pagina viene utilizzato come indice nella *page table* del processo correntemente in esecuzione: ogni entry della page table contiene l'indirizzo di base del frame fisico f che contiene la pagina in questione. Tale indirizzo di base f viene combinato (giustapposto come prefisso) con lo spiazzamento d per definire l'indirizzo fisico da inviare all'unità di memoria:



3. Supponendo di avere un sistema con tre frame e sei pagine, addottando una politica di rimpiazzamento LRU, quanti page fault si verificheranno con la reference string seguente?

   0 1 5 2 3 2 5 1 2 4 1

   (Si assuma che i tre frame siano inizialmente vuoti.)

   Risposta: Si generano 7 page fault:

	0	1	5	2	3	2	5	1	2	4	1
		0	1	5	2	3	2	5	1	2	4
			0	1	5	5	3	2	5	1	2
				0	1	1	1	3	3	5	5
					0	0	0	0	0	3	3
										0	0
	P	P	P	P	P			P		P	

4. Relativamente all'allocazione dello spazio su disco, si illustri brevemente il metodo dell'allocazione concatenata.

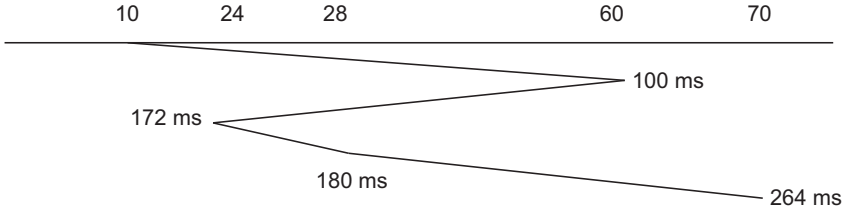
   Risposta: Con l'allocazione concatenata ogni file è costituito da una lista concatenata di blocchi che possono essere sparsi dovunque nel disco. Ogni entry di directory contiene un puntatore al primo ed all'ultimo blocco del file in questione. Ogni blocco contiene un puntatore al blocco successivo. Quindi, volendo aggiungere dei dati ad un file, è sufficiente determinare un blocco libero tramite il sistema di gestione dello spazio libero su disco e concatenare il blocco alla fine del file, aggiornando i puntatori. In questo modo i file possono crescere fintanto che c'è dello spazio libero sul disco. Gli svantaggi sono che parte dello spazio di ogni blocco deve essere dedicato alla memorizzazione del puntatore al blocco successivo e che il sistema risulta efficiente solo per accessi di tipo sequenziale (in quanto, per accedere al blocco i-esimo, bisogna seguire la catena di puntatori). La traduzione da indirizzo logico (LA) ad indirizzo fisico si effettua quindi nel modo seguente: si divide LA per la dimensione del blocco diminuita dallo spazio occupato dal puntatore al blocco successivo. Il quoziente rappresenta la posizione occupata dal blocco nella lista concatenata, mentre il resto incrementato di uno rappresenta l'offset all'interno del blocco stesso.

5. Si consideri un disco gestito con politica C-LOOK. Inizialmente la testina è posizionata sul cilindro 10, ascendente; lo spostamento ad una traccia adiacente richiede 2 ms. Al driver di tale disco arrivano richieste per i cilindri 60, 24, 28, 70, rispettivamente agli istanti 0 ms, 30 ms 40 ms, 178 ms. Si trascuri il tempo di latenza.

   1. In quale ordine vengono servite le richieste?
   2. Il tempo di attesa di una richiesta è il tempo che intercorre dal momento in cui è sottoposta al driver a quando viene effettivamente servita. Qual è il tempo di attesa medio per le quattro richieste in oggetto?

   Risposta:

   1. Le richieste vengono servite in quest'ordine: 60, 24, 28, 70 come evidenziato nel seguente diagramma:

   2. Il tempo di attesa medio è dato da $\frac{(100-0)+(172-30)+(180-40)+(264-178)}{4} = \frac{100+142+140+86}{4} = 117\ ms$.

6. Qual è la differenza fondamentale fra un'architettura UMA ed un'architettura NUMA?

   Risposta: Un sistema multiprocessore UMA (Uniform Memory Access) è n sistema strettamente accoppiato che gode della proprietà che il tempo di accesso a qualunque locazione di memoria è sempre lo stesso (indipendentemente dal fatto che si faccia riferimento a memoria locale o di un altro nodo del sistema). Invece un sistema NUMA (Non Uniform Memory Access), pur rimanendo un sistema strettamente accoppiato, è caratterizzato dal fatto che l'accesso alla memoria locale è più veloce (da 2 a 15 volte) dell'accesso alla memoria remota. Ciò consente di ottenere sistemi scalabili con un gran numero di CPU.

7. Qual è la differenza fondamentale fra servizi di rete e servizi distribuiti?

   Risposta: I servizi di rete offrono ai processi le funzionalità necessarie per stabilire e gestire le comunicazioni tra i nodi di un sistema distribuito (es.: l'interfaccia fornita dalle socket). In sostanza gli utenti devono essere consapevoli della struttura del sistema e devono indirizzare esplicitamente le singole macchine. I servizi distribuiti invece sono modelli comuni (paradigmi di comunicazione) trasparenti che offrono ai processi una visione uniforme, unitaria del sistema distribuito stesso (es: file system remoto). I servizi distribuiti vanno quindi a formare il cosiddetto *middleware*, ovvero, uno strato software fra il sistema operativo e le applicazioni che uniforma la visione dell'intero sistema.

Sistemi Operativi
22 marzo 2007
Compitino 2/B

Si risponda ai seguenti quesiti, giustificando le risposte.

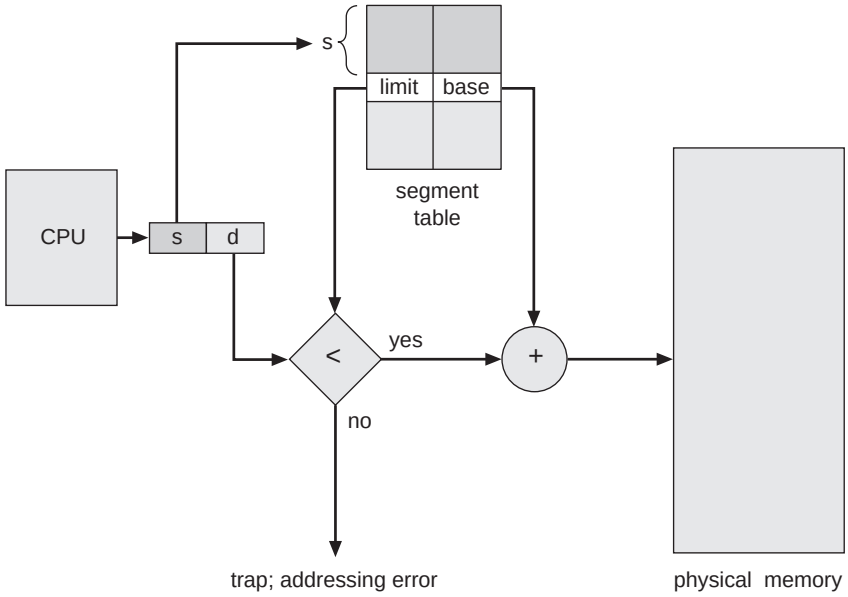
1. Si illustrino i concetti di frammentazione interna ed esterna.

   Risposta: Quando la memoria viene allocata in blocchi di dimensione fissata, con l'espressione frammentazione interna si intende la differenza fra la memoria assegnata ad un processo e quella effettivamente richiesta da quest'ultimo. Quando si alloca la memoria in blocchi di dimensione variabile e si caricano e si rimuovono da quest'ultima dei processi, lo spazio libero si frammenta in piccole parti; si parla di frammentazione esterna quando lo spazio libero complessivo nella memoria è sufficiente per soddisfare una richiesta, ma non è contiguo.

2. Si illustri brevemente il meccanismo di segmentazione della memoria con il relativo schema di traduzione dell'indirizzo virtuale in indirizzo fisico.

   Risposta: La segmentazione è uno schema di gestione della memoria che supporta la visione utente, ovvero, considera un programma come una collezione di segmenti. Un segmento è un'unità logica di memoria; ad esempio: programma principale, procedure, funzioni, variabili locali, variabili globali, stack, tabella dei simboli, memoria condivisa possono essere considerati segmenti separati, di dimensioni differenti. Può soffrire di frammentazione esterna.

   Per ogni processo la segment table di un processo consente di tener traccia dei vari segmenti allocati e di tradurre gli indirizzi logici della forma $\langle segmento, offset \rangle$ nei corrispondenti indirizzi fisici. Il *segmento* funge da indice nella segment table consentendo di individuare l'entry corretta. Quest'ultima è una coppia ordinata: $\langle base, limite \rangle$, dove *base* indica l'indirizzo fisico iniziale dove risiede il segmento e *limite* denota la sua lunghezza. Se $offset$ è minore di *limite*, allora l'indirizzo fisico è determinato dalla somma di *base* e $offset$. In caso contrario viene generata una trap al sistema operativo:



3. Supponendo di avere un sistema con quattro frame e sei pagine, addottando una politica di rimpiazzamento LRU, quanti page fault si verificheranno con la reference string seguente?

   1 0 5 2 3 2 1 5 2 4 1

   (Si assuma che i quattro frame siano inizialmente vuoti.)

Risposta: Si generano 7 page fault:

	1	0	5	2	3	2	1	5	2	4	1
		1	0	5	2	3	2	1	5	2	4
			1	0	5	5	3	2	1	5	2
				1	0	0	5	3	3	1	5
					1	1	0	0	0	3	3
										0	0
	P	P	P	P	P		P			P	

4. Relativamente all'allocazione dello spazio su disco, si illustri brevemente il metodo dell'allocazione indicizzata.

   Risposta: L'allocazione indicizzata raggruppa tutti i puntatori ai blocchi (sparsi nel disco) di un file in un *blocco indice*. Quindi l'i-esimo elemento di tale blocco è un puntatore al blocco i-esimo del file ed è sufficiente mantenere nella directory il puntatore al blocco indice per accedere ai blocchi dati del file. Nel caso di un file vuoto (ad esempio appena creato) tutti i puntatori del blocco indice saranno inizializzati con un valore speciale che non punta ad alcun blocco dati sul disco (solitamente `-1`). Al momento di allocare un nuovo blocco dati per il file è sufficiente aggiornare il primo puntatore "libero" con l'indirizzo del nuovo blocco restituito dal sistema di gestione dello spazio libero su disco. La traduzione da indirizzo logico (LA) ad indirizzo fisico avviene dividendo LA per la dimensione del blocco: il quoziente rappresenta l'offset nel blocco indice, mentre il resto rappresenta l'offset nel blocco dei dati indicato dall'indice. Per file di grosse dimensioni un singolo blocco indice può rivelarsi insufficiente; quindi ci sono due varianti:

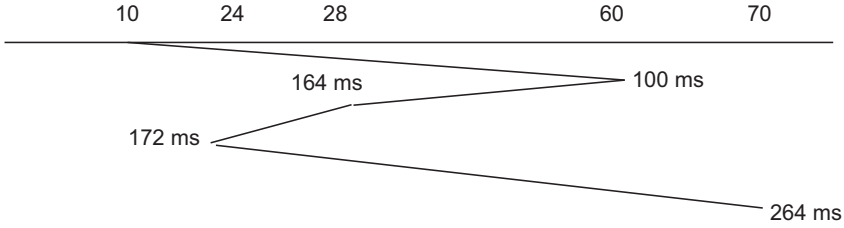
   1. schema concatenato: l'ultimo indirizzo del blocco indice è un puntatore ad un ulteriore blocco indice e così via fin quando è necessario;
   2. indice a più livelli: ogni indirizzo del blocco indice punta a un blocco indice di secondo livello che punta ai blocchi dati (volendo si possono inserire più livelli).

5. Si consideri un disco gestito con politica LOOK. Inizialmente la testina è posizionata sul cilindro 10, ascendente; lo spostamento ad una traccia adiacente richiede 2 ms. Al driver di tale disco arrivano richieste per i cilindri 60, 24, 28, 70, rispettivamente agli istanti 0 ms, 30 ms 40 ms, 120 ms. Si trascuri il tempo di latenza.

   1. In quale ordine vengono servite le richieste?
   2. Il tempo di attesa di una richiesta è il tempo che intercorre dal momento in cui è sottoposta al driver a quando viene effettivamente servita. Qual è il tempo di attesa medio per le quattro richieste in oggetto?

   Risposta:

   1. Le richieste vengono servite in quest'ordine: 60, 28, 24, 70 come evidenziato nel seguente diagramma:

   2. Il tempo di attesa medio è dato da $\frac{(100-0)+(172-30)+(164-40)+(264-120)}{4} = \frac{100+142+124+144}{4} = 127,5\ ms$.

6. Cos'è un *multicomputer*? Si tratta di un sistema strettamente o debolmente accoppiato?

   Risposta: Un *multicomputer* è un sistema di calcolatori strettamente accoppiati, ma senza memoria condivisa. La comunicazione avviene tramite il paradigma dello scambio di messaggi su linee dedicate ad alta velocità. I vari nodi sono costituiti da CPU, RAM, una scheda di rete ad alta velocità ed in

alcuni casi un disco. Il resto delle periferiche sono condivise. Tutti i nodi solitamente sono situati all'interno di una stanza (spesso in armadi appositi) ed ognuno fa girare la propria copia dello stesso sistema operativo.

7. Qual è la differenza fra servizi orientati alla connessione e servizi non orientati alla connessione?

   Risposta: I servizi orientati alla connessione stabiliscono una connessione, ovvero, un canale virtuale per tutta la durata della comunicazione in modo da assicurare il trasferimento dei dati in sequenza. Invece i servizi non orientati alla connessione trasferiscono singoli messaggi senza stabilire e mantenere una vera connessione. In particolare quindi il ricevente può ricevere i messaggi in ordine diverso da quello in cui sono stati spediti.