Sistemi Operativi
28 marzo 2008
Esame completo

Si risponda ai seguenti quesiti, giustificando le risposte.

1. (a) Si descrivano i vari tipi di struttura di un sistema operativo e si diano esempi per ciascun tipo.
   (b) Che cosa si intende per separazione tra meccanismi e politiche?

   Risposta:

   (a) La struttura più semplice è quella fornita da un sistema operativo come MS-DOS (pensato per fornire le massime funzionalità nel minore spazio possibile). Tale sistema, nonostante presenti un po' di struttura, non è chiaramente diviso in moduli e le sue interfacce e livelli funzionali non sono ben separati. Infatti il kernel del sistema operativo può essere tranquillamente "bypassato" dai programmi che possono accedere direttamente all'hardware del calcolatore.

   Anche la versione iniziale di UNIX presentava una strutturazione molto semplice in due sole parti separate: programmi di sistema e kernel (implementava file system, scheduling di breve termine, gestione della memoria ecc.).

   Nell'approccio stratificato invece il sistema operativo è diviso in un certo numero di strati (livelli) dove ogni strato è costruito su quelli inferiori. Lo strato di base (livello 0) è l'hardware; il più alto è l'interfaccia utente. Secondo la modularità, gli strati sono pensati in modo tale che ognuno utilizzi funzionalità (operazioni) e servizi solamente di strati inferiori (es.: THE OS, Linux, Solaris, OS/2).

   Tale approccio viene portato all'estremo dal concetto di macchina virtuale che tratta l'hardware ed il sistema operativo come se fosse tutto hardware. Una macchina virtuale fornisce quindi un'interfaccia identica all'hardware di base sottostante. Il sistema operativo impiega le risorse del calcolatore fisico per creare le macchine virtuali:

   - lo scheduling della CPU crea l'illusione che ogni processo abbia il suo processore dedicato;
   - la gestione della memoria crea l'illusione di una memoria virtuale per ogni processo;
   - lo spooling può implementare delle stampanti virtuali;
   - lo spazio disco può essere impiegato per creare "dischi virtuali";

   Questo approccio è seguito in molti sistemi: Windows, Linux, MacOS,...

   (b) Con l'espressione separazione tra meccanismi e politiche nell'ambito dei sistemi operativi si intende la distinzione fra come eseguire qualcosa (meccanismo) e che cosa si debba fare, ovvero quali scelte operare, in risposta ad un certo evento (politica).

2. Si descrivano i thread di Solaris.

   Risposta: Solaris fino alla versione precedente alla 9 prevede un modello di thread a due livelli in cui più thread a livello utente sono messi in corrispondenza con un numero minore o uguale di thread a livello kernel oppure è possibile vincolare un thread a livello utente ad un solo thread a livello kernel. In questo modo un programmatore può creare a livello utente diversi thread ed il sistema operativo, a seconda del numero di CPU disponibili, può mappare questi ultimi su uno o più thread a livello kernel in modo da garantire una reale concorrenza. Quindi vengono combinati i vantaggi derivanti dai thread a livello utente e da quelli a livello kernel. I primi sono convenienti per processi CPU-bound, ovvero, per processi che non necessitino di un'attività di I/O intensiva (che bloccherebbe il processo con tutti i suoi thread) in cui sia possibile scomporre e parallelizzare il lavoro. I secondi invece sono convenienti per processi I/O-bound in cui è prevalente l'attività di I/O.

   A partire dalla versione 9 Solaris ha adottato il modello uno a uno.

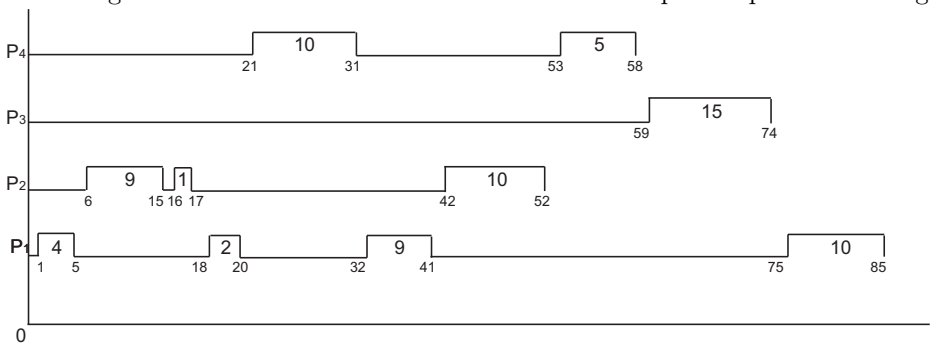
3. Si consideri un sistema con scheduling della CPU a priorità con tre code, A, B, C, di priorità decrescente, con prelazione tra code. Le code A e B sono round robin con quanto di 10 e 15 ms, rispettivamente; la coda C è FCFS. Se un processo nella coda A o B consuma il suo quanto di tempo, viene spostato in fondo alla coda B o C, rispettivamente.

   Nelle code A, B, C entrano i seguenti processi:

	coda	arrivo	burst
P_1	B	0	25ms
P_2	A	5	20ms
P_3	C	15	15ms
P_4	A	20	15ms

Si determini il diagramma di GANTT relativo all'esecuzione dei quattro processi, assumendo che il tempo di latenza del kernel sia pari a 1 ms.

Risposta: Il diagramma di GANTT relativo all'esecuzione dei quattro processi è il seguente:



4. Si spieghi *brevemente* come funzionano i seguenti schemi di allocazione della memoria:
   1. allocazione contigua:
      (a) con partizionamento fisso,
      (b) con partizionamento dinamico,
   2. allocazione non contigua:
      (a) paginazione,
      (b) segmentazione.

   Risposta: Nel caso dell'allocazione contigua la memoria assegnata ad un processo non può essere suddivisa in più parti, ma deve estendersi senza interruzioni da un indirizzo minimo ad uno massimo. Se il partizionamento è fisso, la memoria viene inizialmente suddivisa (ad esempio in fase di boot) in varie aree o partizioni di dimensione fissata. Tali aree non possono essere modificate per quanto riguarda le loro dimensioni; quindi si presenterà un fenomeno di frammentazione interna (ovvero, ci sarà un certo "spreco" di memoria internamente ad ogni partizione). Se il partizionamento è dinamico invece, inizialmente vi è un'unica regione di memoria libera. Ogni volta che un processo viene lanciato in esecuzione, esso si vede assegnare dal sistema operativo l'esatto quantitativo di memoria di cui ha bisogno. Non si verificano quindi problemi di frammentazione interna, ma di frammentazione esterna. Infatti, in seguito al lancio di processi (assegnazione di aree di memoria) ed alla loro terminazione (restituzione di aree di memoria al sistema operativo), si possono formare delle zone residue di memoria libera troppo piccole per allocare un nuovo processo, anche se la somma totale di queste ultime risulta sufficiente a soddisfare le esigenze del nuovo processo.

   Con l'allocazione non contigua si rinuncia all'idea di allocare spazio di memoria senza soluzione di continuità per ogni processo: ad un processo viene allocata memoria fisica dovunque essa si trovi. Nel caso della paginazione si divide la memoria fisica in frame, ovvero, blocchi di dimensione fissa (una potenza di 2, tra 512 e 8192 byte) e si divide la memoria logica in pagine, della stessa dimensione. Il sistema operativo tiene traccia dei frame liberi (per eseguire un programma di n pagine, servono n frame liberi in cui caricare il programma). Si imposta una page table per tradurre indirizzi logici in indirizzi fisici. Non esiste frammentazione esterna, ma una ridotta frammentazione interna.

   La segmentazione invece è uno schema di gestione della memoria che supporta la visione utente di quest'ultima. Un programma è una collezione di segmenti (di dimensione variabile), dove ogni segmento è un'unità logica di memoria (ad esempio: programma principale, procedure, funzioni, variabili locali, variabili globali stack, tabella dei simboli memoria condivisa). L'indirizzo logico consiste in un coppia <segment-number, offset> e la segment table mappa gli indirizzi bidimensionali dell'utente negli indirizzi fisici unidimensionali. Come nel caso dell'allocazione contigua con partizionamento dinamico possono esserci dei problemi di frammentazione esterna.

5. Si spieghi cosa si intende con *interruzione precisa*. Se un sistema ha interruzioni imprecise, quali sono le conseguenze per il sistema operativo?

   Risposta:

   (a) Un'interruzione si dice precisa quando gode delle seguenti quattro proprietà:

1. il program counter viene salvato in un posto noto,
2. tutte le istruzioni che precedono quella puntata dal program counter sono state completamente eseguite,
3. nessuna delle istruzioni che seguono quella puntata dal program counter è stata eseguita,
4. lo stato di esecuzione dell'istruzione puntata dal program counter è noto.

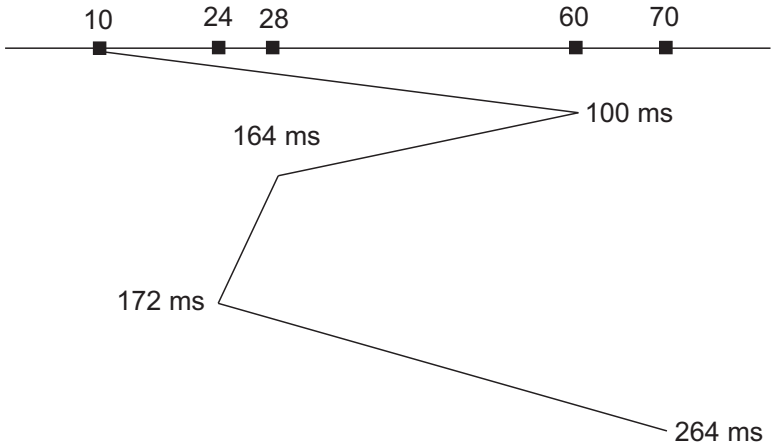
(b) Nel caso in cui si abbiano interruzioni imprecise è difficile riprendere l'esecuzione in modo esatto in hardware: la CPU si limita a "riversare" sullo stack tutta l'informazione relativa allo stato corrente, lasciando che sia il sistema operativo a capire che cosa debba essere fatto. In questo modo rallentano le fasi di ricezione degli interrupt e di context-switch/ripristino dell'esecuzione, provocando grosse latenze.

6. Si consideri un disco gestito con politica LOOK. Inizialmente la testina è posizionata sul cilindro 10, con moto 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 nell'ordine 60, 28, 24, 70, come illustrato dal seguente diagramma:



   2. Il tempo di attesa medio è dato da $\frac{(100-0)+(172-30)+(164-40)+(264-120)}{4} = \frac{100+142+124+144}{4} = \frac{510}{4} = 127,5ms$.

7. Si spieghi *brevemente* cos'è e come funziona una RPC (Remote Procedure Call).

   Risposta: Una Remote Procedure Call (RPC) consente ad un processo in esecuzione su un host di effettuare una chiamata ad una procedura che viene eseguita su un host remoto come se questa fosse una normale chiamata ad una funzione locale. Lo scambio di messaggi necessario per il funzionamento di una RPC è completamente invisibile al programmatore. Sostanzialmente quando un processo su un host A chiama una procedura su un host B, il processo chiamante su A viene sospeso, l'informazione necessaria per la computazione (i parametri della procedura) vengono comunicati via rete e l'esecuzione prosegue su B. Per effettuare una chiamata RPC il processo chiamante (client) utilizza una piccola procedura (client stub) che rappresenta la procedura remota nello spazio di indirizzamento del client. Il client stub organizza i parametri in un messaggio (marshaling) che viene spedito all'host remoto. Giunto a destinazione il messaggio, esso viene elaborato dal server stub che chiama la procedura responsabile della computazione. Il risultato viene poi rispedito via rete dal server stub al client stub che lo restituisce al processo chiamante. Quest'ultimo può quindi riprendere la sua esecuzione come in seguito ad una normale chiamata di procedura locale.

Il punteggio attribuito ai quesiti è il seguente: 2, 2, 3, 5, 3, 3, 4, 3, 3, 4. Totale 32 punti.