Sistemi Operativi
3 settembre 2010
Compito

Si risponda ai seguenti quesiti, giustificando le risposte.

1. (a) Quali sono le componenti comuni ai vari sistemi operativi?
   (b) Quali sono i servizi offerti dai sistemi operativi?
   (c) Si descriva il funzionamento di un sistema operativo basato su microkernel.

   Risposta:

   (a) Le componenti comuni ai vari sistemi operativi sono le seguenti:
      1. gestione dei processi;
      2. gestione della memoria principale;
      3. gestione della memoria secondaria;
      4. gestione dell'I/O;
      5. gestione dei file;
      6. sistemi di protezione;
      7. connessioni di rete (*networking*);
      8. sistema di interpretazione dei comandi.

   (b) I servizi offerti dai sistemi operativi sono i seguenti:
      - esecuzione dei programmi: caricamento dei programmi in memoria e loro esecuzione;
      - operazioni di I/O: il sistema operativo deve fornire un modo per condurre le operazioni di I/O, dato che gli utenti non possono eseguirle direttamente;
      - manipolazione del file system: capacità di creare, cancellare, leggere, scrivere file e directory;
      - comunicazioni: scambio di informazioni tra processi in esecuzione sullo stesso computer o su sistemi diversi collegati da una rete (implementati attraverso *memoria condivisa* o *passaggio di messaggi*);
      - individuazione di errori: garantire una computazione corretta individuando errori nell'hardware della CPU o della memoria, nei dispositivi di I/O, o nei programmi degli utenti.

   (c) In un sistema operativo basato su microkernel, il kernel è ridotto all'osso, fornendo soltanto i meccanismi di comunicazione tra processi, una minima gestione di questi ultimi e della memoria ed una gestione dell'hardware di basso livello (driver). Tutto il resto viene gestito da processi in spazio utente: ad esempio, tutte le politiche di gestione del file system, dello scheduling, della memoria sono implementate come processi. Ciò comporta una minor efficienza rispetto ad un kernel monolitico, ma anche una grande flessibilità ed un'immediata scalabilità in ambiente di rete.

2. (a) Che cosa è un programma di sistema?
   (b) Si spieghi il funzionamento del seguente frammento di codice:

```
while(1) {
   read_command(command, parameters);
   if (fork() != 0) {
     waitpid(-1, &status, 0);
   } else {
     execve(command, parameters, NULL);
   }
}
```

   Risposta:

   (a) I programmi di sistema forniscono un ambiente per lo sviluppo e l'esecuzione dei programmi. Si dividono in programmi per la gestione e per la modifica dei file, per le informazioni sullo stato del sistema e dell'utente, programmi di supporto dei linguaggi di programmazione, di caricamento ed esecuzione dei programmi, per le comunicazioni e per altri scopi (compilatori, elaborazione testi, interprete comandi, ...). I programmi di sistema usano direttamente le chiamate di sistema. In pratica la maggior parte di ciò che un utente vede di un sistema operativo è definito dai programmi di sistema, non dalle reali chiamate di sistema.

(b) Il frammento di codice implementa una semplice shell. Ad ogni iterazione del ciclo infinito viene eseguita la lettura di un comando e dei relativi parametri tramite la funzione `read_command`. Poi, tramite la `fork` viene generato un processo figlio. L'esecuzione del padre si pone in attesa della terminazione del processo figlio per mezzo della chiamata di sistema `waitpid`, mentre il processo figlio (ramo `else`) provvede ad eseguire il comando `command` con i relativi parametri `parameters` tramite la chiamata di sistema `execve`.

3. Si consideri un sistema con scheduling SJF con prelazione (cioè SRTF), ove $\alpha = 0,5$ e $\tau_0 = 20$ msec. All'istante 0 il processore si libera e tre processi, P_1, P_2, P_3, sono in coda ready. Finora il processo P_1 è andato in esecuzione due volte con CPU burst $25, 40$ msec per P_1; mentre P_2, P_3 sono andati in esecuzione una volta con CPU burst di 30 e 60 msec, rispettivamente.
   Si determini:
   (a) Quale processo viene selezionato dallo scheduler all'istante 0?
   (b) All'istante 15 msec entra nella coda ready un nuovo processo P_4 con CPU burst previsto di 20 msec. Il processo selezionato precedentemente è ancora in esecuzione. Che cosa succede?
   (c) Che cosa succede quando il processo in esecuzione termina il suo burst?

   Risposta:
   (a) Per P_1 abbiamo $\tau_1 = \frac{1}{2}\cdot 20 + \frac{1}{2}\cdot 25 = 10 + 12,5 = 22,5$, $\tau_2 = \frac{1}{2}\cdot 22,5 + \frac{1}{2}\cdot 40 = 11,25 + 20 = 31,25$, per P_2 abbiamo $\tau_1 = \frac{1}{2}\cdot 20 + \frac{1}{2}\cdot 30 = 25$ ed infine per P_3 abbiamo $\tau_1 = \frac{1}{2}\cdot 20 + \frac{1}{2}\cdot 60 = 40$. Quindi all'istante 0 SRTF sceglie P_2.
   (b) All'istante 15 msec quando entra nella coda ready un nuovo processo P_4 con CPU burst previsto di 20 msec, continua P_2 perché gli mancano meno di 20 ms ($25 - 15 = 10$ ms) che è il burst previsto per P_4.
   (c) Quando P_2 termina, va in esecuzione P_4.

4. Si descriva la problematica del deadlock e si discutano i diversi approcci utilizzati nei vari sistemi operativi per affrontare tale problematica.

   Risposta: Un insieme di processi si trova in deadlock (stallo) se ogni processo dell'insieme è in attesa di un evento che solo un altro processo dell'insieme può provocare.

   Il problema è caratterizzato da quattro condizioni necessarie (ma non sufficienti):

   1. mutua esclusione: ogni risorsa è assegnata ad un solo processo, oppure è disponibile;
   2. hold&wait: i processi che hanno richiesto ed ottenuto delle risorse, ne possono richiedere altre;
   3. mancanza di prerilascio: le risorse che un processo detiene possono essere rilasciate dal processo solo volontariamente;
   4. catena di attesa circolare di processi: esiste un sottoinsieme di processi {P_0, P_1, ..., P_n} tali che P_i è in attesa di una risorsa che è assegnata a $P_{i+1 \bmod n}$.

   In generale ci sono quattro possibili approcci al problema dello stallo:

   1. ignorare il problema, fingendo che non esista;
   2. permettere che il sistema entri in un deadlock, riconoscerlo e quindi risolverlo;
   3. cercare di evitare dinamicamente le situazioni di stallo, con una accorta gestione delle risorse;
   4. assicurare che il sistema non possa mai entrare mai in uno stato di deadlock, negando una delle quattro condizioni necessarie.

   Fra i precedenti il terzo approccio è quello più diffuso, dato che assicurare l'assenza di deadlock impone costi (in prestazioni, funzionalità) molto alti e questi ultimi non sono ammissibili in molti contesti. Quindi, considerando il rapporto costo/benefici, se la probabilità che accada un deadlock è sufficientemente bassa, non si giustifica il costo per evitarlo. Esempi tipici sono la system call `fork` di Unix e la rete Ethernet e tale approccio è quello adottato dalla maggior parte dei sistemi (Unix e Windows compresi).

5. Si descriva la struttura di un Inode in Unix.

   Risposta: Ogni inode contiene:

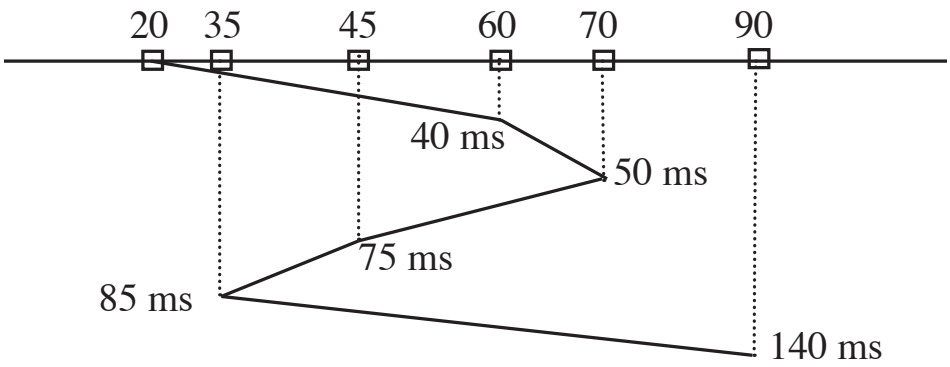
- modo: bit di accesso, di tipo e speciali del file;
- UID e GID del possessore;
- dimensione del file in byte;
- timestamp di ultimo accesso (atime), di ultima modifica (mtime) e di ultimo cambiamento dell'inode (ctime);
- numero di link hard che puntano all'inode;
- blocchi diretti: puntatori ai primi 12 blocchi del file;
- primo indiretto: indirizzo del blocco indice dei primi indiretti;
- secondo indiretto: indirizzo del blocco indice dei secondi indiretti;
- terzo indiretto: indirizzo del blocco indice dei terzi indiretti.

6. Si consideri un disco gestito con politica LOOK. Inizialmente la testina è posizionata sul cilindro 20 con direzione di servizio ascendente; lo spostamento ad una traccia adiacente richiede 1 ms. Al driver di tale disco arrivano richieste per i cilindri 70, 60, 35, 45, 90, rispettivamente agli istanti 0 ms, 15 ms, 20 ms, 45 ms, 55 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 cinque richieste in oggetto?

   Risposta:

   1. Le richieste vengono servite nell'ordine 60, 70, 45, 35, 90:



   2. Il tempo di attesa medio per le cinque richieste in oggetto è $\frac{(40-15)+(50-0)+(75-45)+(85-20)+(140-55)}{5} = \frac{25+50+30+65+85}{5} = \frac{255}{5} = 51\ ms.$