DOMANDE DI TEORIA DI SISTEMI OPERATIVI 1

DOMANDA 1

1) In un sistema operativo un processo a thread singolo può essere nei seguenti stati: Terminated, Waiting, New, Running, Ready. Disegnare un diagramma che illustra quali transizioni sono possibili tra questi 5 stati, e descrivere, per ciascuna di queste transizioni, da quali eventi sono causate.

RISPOSTA 1

1) Inserire lo schema dei vari stati del processo.

New→Ready: il processo viene inizialmente creato, sono allocate le risorse per gestirlo affinché il processo passi allo stato di Ready pronto per essere eseguito.

Ready→Running: il processo viene selezionato dallo scheduler per essere eseguito.

Running→Ready: il processo non ha ancora ancora completato la sua esecuzione, ma ha terminato il time-slice di tempo assegnatogli perciò riceve un time-interrupt e ritorna allo stato Ready per essere nuovamento selezionato dallo scheduler e proseguire in seguito la sua computazione.

Running→Waiting: il processo esegue un'operazione sospensiva (operazione I/O, sleep, wait), interrompe la sua esecuzione rilasciando la CPU e si mette in attesa del completamento della richiesta.

Waiting→Ready: la richiesta sospensiva è terminata, il processo ritorna allo stato Ready.

Waiting→Terminated: in uno qualunque dei tre stati potrebbe ricevere un segnale di interruzione e terminare in modo anomalo; in esecuzione potrebbe eseguire un'operazione non valida e terminare in modo anomalo, oppure completare il suo programma e terminare in modo corretto.

DOMANDA 2

2) Che cosa è il Process Control Block? In un sistema operativo con supporto ai kernel threads, quali informazioni sono per-processo, e quali sono per-thread? Fare un elenco.

RISPOSTA 2

2) Il PCB è una struttura in cui al suo interno sono contenute le informazioni che distinguono un processo dall'altro. Le informazioni contenute all'interno della PCB sono:

- I registri della CPU(PC,SP,PSW,ecc)

- Lo stato del processo(ready, running,waiting,new,terminated)

- I dati per lo scheduling

- i dati per la gestione della memoria (i registri LIMIT e BASE)

- il tempo d’uso della CPU e il tempo di turnaround

- i dati per le informazioni utili per protezione

La PCB di ogni processo è contenuta all’interno della tabella dei processi ed è compito del sistema operativo tenere traccia di tale tabella.

Le informazioni che distinguono un thread da un altro sono il suo stack, il suo stato, i suoi registri della CPU e il PC ovvero il program counter.

Le informazioni che sono comuni a tutti i thread di un processo sono le variabili globali, lo spazio degli indirizzi, i processi figli, gli allarmi, i segnali e delle informazioni relative agli account.

DOMANDA 3

3)Che cosa è un context-switch? In cosa si differenzia il context-switch tra processi rispetto al context-switch tra thread di uno stesso processo?

RISPOSTA 3

Il context-switch è il passaggio dall’esecuzione di un processo/thread corrente P1 all’esecuzione di un altro processo/thread P2. Comporta il salvataggio dello stato corrente della computazione di P1 ed il ripristino di quello di P2 precedentemente salvato. Nel caso di context-switch tra processi devono essere salvate e ripristinate tutte le informazioni elencate nel PCB, mentre nel caso dei thread solo le informazioni che distinguono un thread da un altro (che sono sono il suo stack, il suo stato, i suoi registri della CPU e il PC ovvero il program counter), rendendo tale operazione molto più efficiente.

DOMANDA 4

4) Spiegate in cosa consiste la tecnica della paginazione, specificate la differenza fra indirizzi virtuali e indirizzi fisici e cosa contiene la tabella delle pagine.

RISPOSTA 4

La tecnica della paginazione è un meccanismo per la gestione della memoria virtuale. Per questa tecnica ad ogni processo è assegnato un proprio spazio di indirizzi (virtuali), tale spazio è contiguo e suddiviso in parti di dimensione fissa chiamate pagine (logiche). Nella memoria fisica è disponibile un certo spazio di indirizzi fisici, anch’esso suddiviso in porzioni fisse chiamate frame, su cui verranno copiate al bisogno le pagine logiche. Le pagine logiche e i frame hanno la stessa dimensione. Non tutte le pagine di uno stesso processo devono essere presenti in memoria bensì devono trovarsi in memoria solo le pagine che servono per l’esecuzione.

La tabella delle pagine mantiene il mapping tra pagine virtuali e frame in memoria per un processo ed è indicizzata rispetto al numero di pagine virtuali e per ogni pagina virtuale può corrispondere un frame se la pagina è caricata in memoria altrimenti non corrisponde

nessun frame. Ogni processo ha la sua tabella delle pagine. Inoltre sono presenti dei bit che indicano la presenza/assenza della pagina in RAM, il dirty bit che indica se la pagina è stata modificata ed il referenced bit che indica se si è fatto accesso alla pagina (in lettura o scrittura).

DOMANDA 5

5)Considerate uno spazio di indirizzamento logico di 16 pagine da 1024 byte ed uno spazio di indirizzamento fisico di 64 frames, quanti bit saranno necessari per rappresentare un indirizzo logico e quanti per un indirizzo fisico?

RISPOSTA 5

Per indirizzare 16 pagine logiche servono 4 bit in quanto 24=16, con pagine di dimensione 1024 byte sono necessari 10 bit per indirizzare le diverse locazioni di memoria, di conseguenza la lunghezza totale di un indirizzo logico è 4+10=14 bit. Per l'indirizzo fisico servono 6 bit per indirizzare 64 frame in quanto 26=64, quindi la dimensione di un indirizzo fisico è 6+10=16 bit.

DOMANDA 6

6)E' noto che esiste un algoritmo di rimpiazzamento ottimale delle pagine, ossia che minimizza il numero di page fault rispetto a qualunque altro algoritmo di rimpiazzamento. Spiegate brevemente come funziona.

RISPOSTA 6

L'algoritmo ottimale prevede di scartare la pagina che non verrà usata, o che verrà usata più lontano in futuro rispetto alle altre. Nella pratica non può essere implementato proprio perché necessita di sapere quali pagine verranno usate in futuro.

DOMANDA 7

a) Spiegare che differenza c'è tra chiamate di sistema e funzioni di libreria, e perché esistono le une e le altre.

RISPOSTA 7

a) Le funzioni di libreria esistono per utilizzare codice già scritto; le chiamate di sistema anche e soprattutto per svolgere funzioni che richiedono di accedere a dispositivi, o a dati del sistema operativo, in modo controllato: vi si può accedere solo in modalità kernel, il passaggio è automatico con la chiamata (e all'uscita si ritorna in modalità utente), in questo modo i programmi ottengono quanto desiderato senza dover conoscere come comandare dispositivi o come sono strutturati i dati del sistema operativo, e senza poterli comandare / modificare a piacimento.

DOMANDA 8

b) Il sistema operativo svolge delle funzioni ma consuma anche risorse (spazio e tempo) del sistema di elaborazione. Farne degli esempi relativi alla i) gestione dei processi e alla ii) gestione della memoria, evidenziando che cosa bisogna evitare, o tenere sotto controllo, e come.

RISPOSTA 8

b) Per esempio il S.O. consuma tempo nella commutazione di contesto, che quindi non deve essere troppo frequente (se ne deve tenere conto nella scelta del quanto di tempo); consuma spazio per la tabella dei processi e soprattutto per la tabella delle pagine (che non deve quindi essere allocata per tutto lo spazio di indirizzi potenziale del processo, ad esempio utilizzando più livelli); consuma tempo nella traduzione degli indirizzi specie con la paginazione, tale traduzione viene velocizzata con il TLB; con la paginazione su richiesta il tempo di accesso medio alla memoria rischia di salire molto se i page fault non sono molto infrequenti, va usato un algoritmo di rimpiazzamento che faccia buone scelte (ma non consumi troppo tempo nella scelta) e il tasso di page fault va controllato, eventualmente facendo swap out di uno o più processi, per evitare il thrashing.

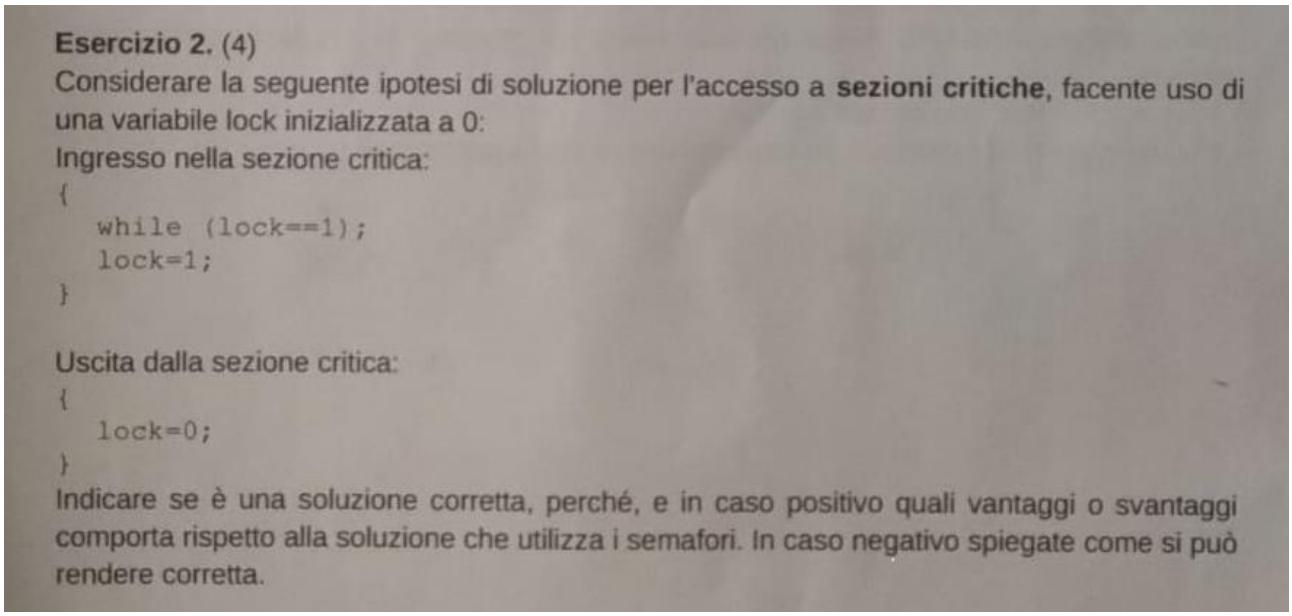
DOMANDA 9

Illustrare perché nei sistemi timesharing è opportuno prevedere la transizione dallo stato running (in esecuzione) allo stato ready (pronto per l'esecuzione) e in conseguenza di quale evento e con quale meccanismo si verifica.

RISPOSTA 9

Il meccanismo che causa il passaggio dallo stato running allo stato di ready di un processo è lo scadere del quanto di tempo che era stato assegnato al processo al momento del suo ingresso nello stato di running e quindi il tempo in cui poteva usare la CPU. Una volta che è scaduto il quanto di tempo il processo che era nello stato di running ritorna in fondo alla coda dei processi ready in attesa di poter riutilizzare la CPU. Inoltre è importante prevedere questa transizione poiché bisogna impedire che un processo monopolizzi la CPU ovvero occupi per troppo tempo la CPU che è inoltre anche l’obiettivo per cui nascono i sistemi timesharing.

DOMANDA 10



Esercizio 2. (4)
Considerare la seguente ipotesi di soluzione per l'accesso a **sezioni critiche**, facente uso di una variabile lock inizializzata a 0:
Ingresso nella sezione critica:

```
{
   while (lock==1);
   lock=1;
}
```

Uscita dalla sezione critica:

```
{
   lock=0;
}
```

Indicare se è una soluzione corretta, perché, e in caso positivo quali vantaggi o svantaggi comporta rispetto alla soluzione che utilizza i semafori. In caso negativo spiegate come si può rendere corretta.

RISPOSTA 10

In questa soluzione software abbiamo una variabile condivisa lock inizializzata a 0. Quando un processo vuole entrare in regione critica prima controlla la variabile lock, se è a 0 significa che nessun processo e in zona critica e allora può entrarci, altrimenti se lock è a 1 significa che c'è già un processo in regione critica e non si può entrare in regione critica.

Il problema di questa soluzione è che supponiamo che un processo legga il lock e veda che è a 0 e quindi entri in regione critica ma, prima che riesca ad impostarlo ad 1 viene scelto dallo scheduler un altro processo che va in running e che vuole entra nella stessa regione critica del primo processo. Anche lui vedrà che lock sarà a 0 poiché il primo processo non è riuscito ad impostare lock a 1 e quindi entrerà in regione critica e imposterà lock a 1. In questo modo avremmo due processi in regione critica e non viene rispettata la mutua esclusione.

Per far rispettare la mutua esclusione dovremmo introdurre il blocco del bus di memoria attraverso l'istruzione TSL così facendo rendiamo possibile la lettura della variabile lock solamente da un processo alla volta e così garantiamo la mutua esclusione.

DOMANDA 11

Illustrare che cosa si intende per multiprogrammazione, per che motivo la si è introdotta (in particolare, quale vantaggio offre) e quali altri servizi sono richiesti al sistema operativo a causa della sua introduzione.

RISPOSTA 11

Per multiprogrammazione si intende la presenza di più programmi caricati in memoria, almeno parzialmente. In questo modo si può passare dall'esecuzione di un processo p1 a quella di un altro p2 in un tempo sufficientemente basso (nel caso in cui p1 debba attendere una operazione di I/O, in un tempo sensibilmente più basso di quello necessario per completare l'operazione); così si può sfruttare meglio la CPU facendo girare p2 nel frattempo. É stata introdotta proprio per migliorare l'utilizzazione della CPU. Richiede la gestione di questi passaggi (context switch, in teoria possibile anche caricando e scaricando l'immagine del processo da disco, ma molto meno conveniente), la traduzione degli indirizzi e la protezione di un processo da accessi accidentali o voluti da parte di un altro.

DOMANDA 12

All'interno di una operazione down su un semaforo, un processo può venire sospeso. Allora in che senso si dice che l'operazione deve essere atomica (indivisibile)? Spiegare che cosa in effetti non deve essere diviso, con particolare riferimento al caso di due processi che chiamano (quasi) contemporaneamente down sullo stesso semaforo. Quali meccanismi si possono usare per ottenere l'atomicità per l'esecuzione di down e up e perché questi meccanismi non sono una buona soluzione generale al problema delle "corse critiche" ma sono accettabili per l'implementazione di down e up?

RISPOSTA 12

Non deve essere possibile, nel caso in cui il semaforo vale 1, che due processi trovino entrambi il valore del semaforo >0 prima che entrambi lo decrementino (quindi, almeno per uno dei due, test e decremento sono «divisi» da operazioni dell'altro) procedendo quindi senza sospendersi. Le operazioni possono essere realizzate come chiamate di sistema; su sistemi uniprocessore l'atomicità può essere realizzata disabilitando le interruzioni (accettabile perché fatto all'interno di una chiamata di sistema e non lasciandole disabilitare al programmatore); su sistemi multiprocessore con istruzioni come TSL, con attesa attiva, accettabile perché si deve attendere l'esecuzione di poche istruzioni macchina.

DOMANDA 13

Commentare il seguente frammento di programma, posto prima di un'operazione di lettura in una delle soluzioni viste nel corso per il problema dei lettori e scrittori indicando anche il valore a cui dovranno essere inizializzati i semafori, e in che caso la seconda operazione down può risultare sospensiva:

```
down(&mutex);

rc=rc+1;

if(rc==1)down(&db);

up(&mutex);
```

Indicare se lo si ritiene utile i requisiti di sincronizzazione: possono accedere al database diverse operazioni di lettura o una operazione di scrittura.

RISPOSTA 13

Con la down su mutex (inizializzato a 1) si prende l'accesso esclusivo alla variabile rc (contatore delle operazioni di lettura in corso), la si incrementa, se vale 1 non ci sono altre operazioni di lettura in corso, quindi bisogna prendere con la down su db (inizializzato a 1) l'accesso esclusivo al database; l'operazione può risultare sospensiva se c'è una scrittura in corso. In ogni caso (down non chiamata, o superata) si rilascia l'accesso esclusivo a rc.

DOMANDA 14

Illustrare se e come con la paginazione si ottiene, o si può ottenere, uno dei possibili scopi del sistema di gestione della memoria: proteggere un processo dagli altri, per evitare in particolare che un processo possa modificare i dati di un altro processo (eccetto in caso di esplicita richiesta di condividere memoria).

RISPOSTA 14

Con la paginazione i dati di un processo A sono protetti da accessi (per errore o per malizia) da parte di un processo B –tranne nel caso in cui il codice dei processi sia progettato per condividere parte dei dati, con esplicita richiesta al S.O.-perché la traduzione degli indirizzi fa sì che B possa accedere soltanto ai frame della RAM che sono stati allocati a B. Questo perché la tabella delle pagine è gestita dal sistema operativo; usando, a partire da un indirizzo di pagina fisica assegnata a B, un offset compreso fra 0 e 2n-1 (dove n bit sono usati per l'offset), si arriva a un indirizzo fisico di una pagina di B. Se B tenta di accedere ad un proprio indirizzo virtuale che non corrisponde a spazio allocato al processo (una pagina per la quale, ad esempio con una tabella delle pagine a più livelli, non c'è un elemento della tabella), si ha un errore.