Multiprogrammazione e multitasking non sono sinonimi per rappresentare lo stesso concetto: V

La multiprogrammazione offre un ambiente in cui le risorse del sistema sono impiegate in modo efficiente: i processi caricati nella RAM vengono eseguiti dalla CPU senza un sistema d'interazione con l'utente.

Il multitasking (aka timesharing), invece, consente la partizione del tempo d'elaborazione: la CPU esegue più lavori (job) commutando le loro esecuzioni con una frequenza tale da permettere a ciascun utente l'interazione con il programma durante l'esecuzione, facendo sembrare che i processi siano eseguiti in parallelo.

Gli obiettivi fondamentali di un sistema operativo sono astrazione ed efficienza: V

Astrazione: un sistema operativo deve semplificare il più possibile l'uso del sistema

Efficienza: un sistema operativo deve essere il più efficiente possibile per permettere l'astrazione del sistema.

L'automatic job sequenceng è un modulo dei moderni sistemi operativi per gestire i processi: F

Nella 2° Generazione, c'era un grande spreco della CPU nel caricare i job nel sistema e riavviare l'elaboratore. Si crè quindi l'automatic job sequencing, ossia un programma chiamato "monitor residente" (nella memoria), che si occupava del trasferimento automatico dell'elaboratore da un job all'altro. All'accensione del calcolatore si avviava il monitor, che trasfriva il controllo a un job; quando esso finiva ritornava il controllo al monitor che faceva eseguire un altro job. In questo modo sequenzializzava i job.

Il DMA è la parte di memoria fisica che viene usata dalla CPU: F

Il DMA (Direct Memory Access) è un meccanismo asincrono che permette la sovrapposizione di CPU e I/O sulla stessa macchina.

Tutte le operazioni di I/O sono eseguite in modalità protetta: V

Per proteggere l'I/O, tutte le operazioni sono privilegiate: Solo il SO può accedere alle risorse I/O: le istruzioni per l'accesso all'I/O invocano delle SYSCALL, (un interrupt sw che cambia la modalità da user a supervisor), solo al termine ritorna la modalità user.

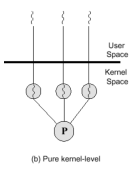
Un processo durante l'esecuzione può essere interrotto solo quando esegue operazioni di I/O: F

Nei moderni SO, un processo può essere interrotto anche da uno scheduler preemptive, o da un interrupt.

I moduli del kernel di un sistema operativo sono eseguiti sempre come processi utente ma in modalità protetta: F

I moduli del kernel possono essere eseguiti in modo diverso in base al sistema adottato.

Se si usano thread a livello kernel il blocco di una thread blocca l'intero processo: F



A livello kernel, lo scheduling è sui thread e non sui processi. Questo vuol dire che se un thread si blocca (per colpa dello scheduler o altro) viene messo in esecuzione un altro thread. Ciò non blocca il processo dei thread. Come si può vedere nella foto il kernel vede i thread e quindi può gestirli senza bloccare il processo chiamante.

Le thread in cui un processo può essere suddiviso condividono lo stesso spazio di indirizzamento: V

I thread appartenenti ad un processo, condividono il PCB del processo (NB. OGNI THREAD HA IL SUO TCB NON CONDIVISO), e lo spazio di indirizzamento del processo. I thread avranno i propri User Stack e Kernel Stack.

L'immagine in memoria di un processo è costituita dalle sezioni PCB, Stack, Codice, Dati, Variabili, Istruzioni: F

Istruzioni e Codice sono la stessa cosa, inoltre manca l'HEAP. Il processo, quindi, ha come immagine: PCB, STACK, DATI, HEAP, CODICE.

Affinchè si verifichi un deadlock è sufficiente che sia vera almeno una delle seguenti condizioni: mutua esclusione, possesso e attesa, non prelazione e attesa circolare : da vedere a lezione

---altro---

PARTE 1

1) Time sharing è la stessa cosa di multitasking? VERO

   Si, è un suo sinonimo. Il multitasking, difatti, spezza i processi in piccole unità di tempo in modo da farli sembrare in parallelo.

2) JVM è sistema monolitico? FALSO
   La JVM è un sistema virtualizzato

3) Le variabili globali sono salvate nello stack? FALSO
   Le variabili globali sono salvati nella parte "DATI". Quelle locali sono slalvate nello stack.

4) Lo stack dell'immagine del processo è eseguito in modalità kernel?

5) Skippata

6) Due processi indipendenti sono deterministici e non riproducibili? FALSO

   Un processo si piò riprodurre, vedi la fork

7) Possono accedere due processi alla mutua esclusione? FALSO
   Non possono accedere alla sezione critica in due

8) La VM favorisce il timesharing?
   Si! Perché il timesharing è pensato per offrire un sistema a time sharing multiplo

9) Semafori: Sincronizzati Si, non sincronizzati NO
10) La VM permette il time sharing? Si
13) 32 bit sono sufficienti per indirizzare 4gb di memoria? SI
14) Il buffering risolve il problema dello spooling? No! È l'incontrario
15) Se sei blocca una thread a livello utente, blocca tutto il processo? Vero. Perché il kernel non riesce a distinguere i diversi kernel.
17) I sistemi batch puntano più alla mole di lavoro che al tempo di risposta? Vero. Solo dalla seconda generazione si introduce il migliorare il tempo
18) I sistemi batch sono stati i primi a introdurre la multiprogrammazione? Falso. La multipogramamzione si introduce solo nella terza generazione