Sistemi operativi 1 – 6/2/17

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

**A:** ad 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.

**Spiegare che differenza c’è tra chiamate di sistema e funzioni di libreria, e perché esistono le une e le altre.**

**B:** 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 i dispositivi o come sono strutturati i dati del sistema operativo, e senza poterli comandare / modificare a piacimento.

**Che relazione ci può essere tra l’invio di un segnale in Unix e una chiamata di sistema? E tra l’invio di un segnale e una *trap*?**

**C:** tutti i segnali possono essere inviati con la chiamata *kill, questo mette in realzione una chiamata di sistema e un segnale di invio su Unix.* Alcuni segnali (SIGILL, SIGSEGV) vengono inviati in conseguenza di una *trap.*

**Descrivere quali tipi di eventi possono far passare un processo o thread nello stato di attesa (waiting), partendo da quali altri stati; e quali eventi possono farlo uscire da tale stato, indicando qual è lo stato successivo.**

**D:** dallo stato “in esecuzione” si passa allo stato di attesa effettuando (direttamente con una chiamata di sistema o attraverso funzioni di libreria) una richiesta di I/O, una chiamata di sistema per l’attesa di un processo figlio, (direttamente o no) una chiamata di sistema di sincronizzazione (*down* su un semaforo, *wait* su condizione). Si esce passando allo stato “pronto” rispettivamente quando: una interruzione hardware segnala che l’operazione di I/O è terminata, il processo figlio termina, avviene la chiamata di sincronizzazione corrispondente (*up* su semaforo, *signal* su condizione)

**Spiegare la differenza tra semafori binari e semafori contatore. Quali di essi sono forniti in Unix con le chiamate *semget, semop*?**

I semafori binari sono semafori che possono avere come valori solo 0 e 1. Generalmente vengono inizializzati a 1 e poi gestiti tramite operazioni di UP e DOWN che corrispondo alle operazioni per incremento e decremento del semaforo. Se un processo vuole entrare in SC e fa una down, se il semaforo vale 1, lo decrementa ed entra in SC, sennò viene sospeso in attesa di una UP. Un semaforo contatore può assumere qualsiasi valore >=0. Può essere usato per il problema di sincronizzazione con un numero N di risorse da assegnare.

**Data una soluzione a un problema di sincronizzazione che utilizza (anche) i semafori contatore, può essere realizzata disponendo solo dei semafori binari? Come?**

**E:** Si, perché i semafori contatore non sono altro che una generalizzazione dei semafori binari. Infatti si può scrivere un semaforo contatore come struttura di semafori binari in questo modo:

typedef struct{

int c,sospesi;

binsem s1,s2;

}sem;

dove c è il valore del semaforo, sospesi il num. Di processi sospesi , binsem s1 per la mutua esclusione su c e sospesi, s2 per aspettare se c non è >0.

**Spiegare, motivando le risposte, che cosa rende necessaria la distinzione tra indirizzi logici e indirizzi fisici, quando e come potrebbe avvenire la traduzione dagli uni agli altri, quando è più opportuno che avvenga. Illustrare come avviene nel caso della paginazione evidenziando il supporto fornito dall’*hardware*.**

**F:** la distinzione è necessaria per avere in memoria più programmi (multiprogrammazione) anche allocati interamente, e anche per averli allocati parzialmente nel caso la dimensione di uno solo, o quella totale, sia maggiore della dimensione della RAM.

La traduzione può avvenire a tempo di compilazione (ma il programma andrebbe eseguito sempre allocandogli la stessa parte di RAM), a tempo di caricamento (ma se ad esempio si fa swap out, allo swap in va ricaricato nella stessa parte di RAM, o va rifatta la traduzione degli indirizzi) oppure a run time, che è più flessibile e facilita il confinamento del processo nello spazio ad esso allocato.

La traduzione degli indirizzi logici in indirizzi fisici nel caso della paginazione avviene nel seguente modo:

La cpu invia l’indirizzo virtuale alla MMU. L’hardware controlla dapprima se il relativo numero di pagina virtuale è presente nel TLB confrontandolo simultaneamente con tutti gli elementi. Se si trova un elemento con lo stesso numero di pagina virtuale ( page hit) e non si ha violazione dei bit di protezione, il numero della pagina fisica viene direttamente preso dal TLB senza bisogno di dover accedere alla tabella delle pagine. Se il numero di pagina virtuale risulta presente nel TLB ma l’istruzione sta tentando di scrivere in una pagina a sola lettura, viene generato un errore di protezione, in maniere del tutto analoga a quanto verrebbe fatto in seguito a un accesso alla tabella della pagine. Il caso interessante si ha quando il numero di pagina virtuale non è presente nel TLB. La MMU si accorge della mancanza per via di un page miss ed esegue una normale ricerca sulla tabella delle pagine. In seguito scarica uno degli elementi del TLB e lo rimpiazza con l’elemento della tabella delle pagine appena trovato. Così se a quella pagina si farà riferimento nell’immediato futuro si avrà un page hit. Quando si cancella un elemento dal TLB il bit di modifica viene copiato nella tabella delle pagine in memoria mentre gli altri valori si trovano già nell’elemento che sta in memoria. Quando si carica il TLB dalla tabella delle pagine, tutti i campi vengono presi dalla memoria.

**Descrivere l’algoritmo di rimpiazzamento delle pagine detto “dell’orologio”. Spiegare se, quanto e perché le pagine scelte possono essere considerate buone scelte come pagine da rimpiazzare.**

**G:** L’algoritmo di rimpiazzamento detto dell’orologio è una semplificazione implementativa dell’algoritmo della seconda possibilità. Si usa una lista circolare e anziché portare in fondo alla lista la pagina non rimpiazzata ( perché aveva R=1) si resetta R e si fa avanzare il puntatore. La bontà della scelta dipende dal fatto che una pagina “graziata”, se continua ad essere usata, verrà probabilmente riferita di nuovo prima che l’algoritmo, in una esecuzione successiva, vi arrivi di nuovo, e quindi continuerà a non essere scelta. Quindi (a meno che quanto sopra non tenda a capitare per tutte le pagine) si scelgono pagine meno usate di altre, senza scorrerle tutte.