SISTEMI OPERATIVI – 15 giugno 2018 corso A

Cognome:_____**Nome:**_____
Matricola:_____

Ricordate che non potete usare calcolatrici o materiale didattico, e che potete consegnare al massimo tre prove scritte per anno accademico.

ESERCIZI RELATIVI ALLA PARTE DI TEORIA DEL CORSO

ESERCIZIO 1 (7 punti)

Tre processi P_A, P_B e P_C eseguono il seguente codice:

Shared **Var** semaphore mutex = 1; (valore iniziale)
semaphore done = –1; (valore iniziale)

P_A:	**P_B:**	**P_C:**
repeat forever:	**repeat forever**:	**repeat forever**:
wait(done)	**wait**(done)	**wait**(mutex)
wait(mutex)	**wait**(mutex)	<C>
<A>	<B>	**signal**(mutex)
signal(mutex)	**signal**(mutex)	**signal**(done)
	signal(done)	

a1)
L'esecuzione concorrente di P_A, P_B e P_C produce una sequenza (di lunghezza indefinita) di chiamate alle procedure A, B e C. Quali delle sequenze qui sotto riportate possono essere la porzione iniziale di sequenze prodotte dall'esecuzione concorrente di P_A, P_B e P_C? (marcate la/le sequenza/e che scegliete con una croce nello spazio apposito, e assumete che nella coda di waiting di "done" sia inizialmente addormentato un processo)

1. [ ] C,C,C,A,B,A,A,B,C …
2. [ ] B,C,C,A,B,A,C,B,C…
3. [X] C,C,A,C,A,C,B,C,B…
4. [ ] C,C,C,B,A,B,A,B,C…

b)
Riportate lo pseudocodice della prima "soluzione" al "*problema dei 5 filosofi*" vista a lezione.

filosofo *i*:

```
do{
        wait(bacchetta[i])
        wait(bacchetta[i+1 mod 5])
                …
                mangia
                …
        signal(bacchetta[i]);
```

```
        signal(bacchetta[i+1 mod 5]);
                …
              pensa
                …
        }while (true)
```

semaphore *bacchetta*[*i*] = 1;

c) E' possible che i processi/filosofi della soluzione riportata al punto b) non vadano mai né in deadlock né in starvation? (Motivate le vostre risposte)

Si. Sebbene la soluzione soffra sia di deadlock che di starvation, è possibile che l'ordine con cui i processi vengono mandati in esecuzione e usano le varie risorse sia tale da non produrre mai una situazione di deadlock o starvation

d) Elencate le tecniche che un SO usa per conservare sempre il controllo della macchina anche quando non sta girando.

Uso di istruzioni privilegiate, uso di un timer che restituisce il controllo al SO quando scade, uso di registri speciali per controllare l'accesso ad aree di ram riservate ai vari processi (oppure, paginazione e/o segmentazione)

e) Che vantaggio da l'uso dei thread al posto dei processi?

Un insieme di peer thread condivide lo spazio di indirizzamento. Per questa ragione, il context switch tra peer thread e la creazione di un nuovo peer thread richiedono molto meno tempo delle corrispondenti operazioni sui processi.

ESERCIZIO 2 (7 punti)

In un sistema la memoria fisica è divisa in 2^{16} frame, un indirizzo logico è scritto su 33 bit, e all'interno di una pagina, l'offset massimo è FFF.

a) Quanti byte occupa la la page table più grande del sistema? (motivate numericamente la vostra risposta)

Un frame/pagina è grande 2^{12} = 4096 byte, e quindi la page table più grande può avere $2^{(33-12)} = 2^{21}$ entry. Nel sistema vi sono 2^{16} frame, per cui sono necessari 2 byte per scrivere il numero di un frame, e quindi la page table più grande occupa ($2^{21} \cdot 2$) byte = 4 Mbyte

b) Assumendo che un indirizzo logico sia sempre scritto su 33 bit e che la memoria fisica del sistema sia sempre divisa in 2^{16} frame, quale dimensione minima dovrebbero avere le pagine di questo sistema per essere certi di non dover ricorrere ad una paginazione a più livelli?

Poniamo 33 = m + n (m = bit usati per scrivere un numero di pagina, n = bit usati per scrivere l'offset).
Allora il numero di entry della PT più grande (cioè 2^m entry), moltiplicato per la dimensione di una entry (cioè 16 bit, o due byte) deve poter essere contenuto in una pagina/frame, ossia: $2^m \cdot 2^1 \leq 2^n$
Da cui: $m + 1 \leq n$. Poiché $m = 33 - n$, risolvendo il semplice sistema si ha $n \geq 17$, ossia le pagine devono almeno essere grandi 2^{17} = 128 Kbyte.

c) assumendo che nel sistema possano essere presenti al massimo 256 processi, e facendo riferimento ai dati del problema indicati nel punto a), quanto sarebbe grande la IPT del sistema? (Motivate numericamente la vostra risposta)

La IPT ha un numero di entry pari al numero di frame del sistema, e ogni entry della IPT deve contenere il PID di processo (un byte) e il numero di una pagina (3 byte).
Dunque la IPT è grande $2^{16} \cdot$ (1+3) byte = 256 Kbyte.

d) perché i sistemi operativi moderni non usano l'allocazione contigua dello spazio in RAM a partizioni fisse o a partizioni variabili?

Non usano l'allocazione a partizioni fisse perché limita a priori il grado di programmazione e soffre in modo eccessivo del problema della frammentazione interna. Non usano l'allocazione a partizioni variabili perché soffre del problema della frammentazione esterna e costringe periodicamente al ricompattamento dello spazio in RAM.

e) Qual è la differenza tra un sistema che adotta la paginazione della memoria principale e un sistema che adotta la segmentazione della memoria principale?

Entrambi adottano una allocazione non contigua dello spazio in RAM. Tuttavia, nei sistemi paginati la memoria principale è partizionata in frame di dimensione fissa, e lo spazio di indirizzamento logico è partizionato in pagine della stessa dimensione. Nei sistemi segmentati lo spazio in ram è allocato in partizioni di dimensione variabile, una partizione per ogni segmento in cui è suddiviso lo spazio di indirizzamento di ciascun processo.

ESERCIZIO 3 (6 punti)

a) Considerate la seguente sequenza di comandi Unix (assumete che tutti i comandi lanciati possano essere correttamente eseguiti):

```
1:   cd /tmp
2:   mkdir mynewdir
3:   cd mynewdir
4:   echo "ciao" > pippo        // crea un nuovo file di nome pippo contenente la stringa ciao
5:   ln pippo paperino
6:   ln –s pippo pluto
7:   ln paperino topolino
8:   rm pippo
9:   cat pluto                  // cat stampa il contenuto del file passato come argomento
10:  cd ..
11:  mkdir aseconddir
```

Dopo l'esecuzione di tutti i comandi:
qual è il valore del link counter nell'index-node associato al link fisico *topolino*? 2
qual è il valore del link counter nell'index-node associato al link fisico *mynewdir*? 2
cosa possiamo dire del link counter dell'index-node associato al link fisico *tmp*? Che è aumentato di 2, a causa delle entry ".." inserite dentro le sottocartelle mynewdir e aseconddir.
Qual è l'output del comando numero 9? "no such file or directory"

b) In Unix è più veloce l'accesso ad un file mediante un link fisico o un link simbolico? (motivate la vostra risposta)

mediante un link fisico, perché occorre leggere un index-node in meno.

c) Perché non sono permessi i link fisici tra directory?

Perché, possono crearsi nel file system dei percorsi circolari che non permettono alle visite ricorsive del file system di terminare.

d) In quale caso, e perché, l'allocazione indicizzata dello spazio in memoria secondaria è particolarmente svantaggiosa?

Nel caso di file piccoli, perché per leggere i dati del file occorre sempre leggere prima il blocco indice, e perché il blocco indice rimane quasi completamente inutilizzato.

d) In quali casi, rispettivamente, è meglio usare un sistema RAID nella configurazione 0, 1 o 5?

Usiamo il RAID 0 se abbiamo bisogno di massimizzare lo spazio di memorizzazione disponibile e la velocità di accesso ai dati, mentre l'affidabilità non è un requisito fondamentale.
Usiamo il RAID 1 se abbiamo bisogno della massima affidabilità e velocità di accesso ai dati.
Usiamo il RAID 5 se vogliamo un ragionevole compromesso tra affidabilità, velocità di accesso ai dati, e spazio di memorizzazione disponibile.