SISTEMI OPERATIVI E LABORATORIO
Indirizzo Sistemi e Reti - 21 luglio 2009

Cognome:_____Nome:_____
Matricola:_____

1. Ricordate che non potete usare calcolatrici o materiale didattico, e che se consegnate annullate automaticamente qualsiasi voto conseguito in una prova precedente.
2. Ricordate che se prendete un voto inferiore a 10 in una qualsiasi delle due parti dello scritto, dovete saltare lo scritto di settembre.
3. Gli **studenti in corso che decidono di sostenere solo la parte di teoria o solo la parte di laboratorio** devono consegnare i loro elaborati al termine della prima ora dello scritto.
4. Gli **studenti degli anni precedenti** devono sostenere l'intero scritto.
5. Si ricorda che a partire dallo scritto di settembre non è più possibile sostenere separatamente l'esame delle due parti del corso. Chi non ha conseguito la sufficienza in *entrambe* le parti entro il secondo scritto di luglio, da settembre deve comunque ridare l'intero scritto.

Scelgo di svolgere (solo per chi consegna alla fine della prima ora):
[ ] la parte relativa alla teoria. **[ ] la parte relativa al laboratorio UNIX**

ESERCIZI RELATIVI ALLA PARTE DI TEORIA DEL CORSO

(il punteggio conseguito farà media con quello ottenuto nella parte di laboratorio. E' comunque necessario prendere almeno 18 punti per considerare passata la parte di teoria.)

ESERCIZIO 1 (9 punti)

Quattro processi arrivano al tempo indicato e consumano la quantità di CPU indicata nella tabella sottostante)

Processo	T. di arrivo	Burst
P1	0	13
P2	2	9
P3	2	7
P4	12	4

a) Calcolare il turnaround medio e il waiting time medio per i processi nel caso dell'algoritmo di scheduling SJF preemtpive (shortest remaining time first). RIPORTANDO IL DIAGRAMMA DI GANTT USATO PER IL CALCOLO.

Turnaround medio:
(0)….P1 …(2) …. P3….(9)….P2….(12)….P4….(16)….P2…(22)….P1….(33)
P1 = 33; P2 = 20; P3 = 7; P4 = 4; 64/4 = 16

Waiting time medio:
(basta ricordarsi di sottrarre al turnaround di ogni processo, la durata del suo burst):
P1 = 20; P2 = 11; P3 = 0; P4 = 0; 31/4 = 7,75

b) SJF premptive può causare problemi di starvation? E SJF *non* preemptive?

Si. In entrambi gli algoritmi, potrebbero arrivare in coda di ready sempre nuovi processi con un CPU time inferiore a processi già in coda di ready e in attesa di essere schedulati.

c) Descrivete brevemente la differenza tra un algoritmo di scheduling preemptive e un algoritmo non-preemptive

Si vedano i lucidi della sezione 5.1.3

d) Descrivete brevemente un algoritmo di scheduling a code multiple con retroazione

Usa più code di ready, gestite ciascuna con una diversa politica di scheduling. Un processo può essere spostato da una coda all'altra in base a come si è comportato l'ultima volta che gli è stata assegnata la CPU.

e) Riportate il diagramma di stato della vita di un processo.

Vedere il lucido della sezione 3.1.2.

ESERCIZIO 2 (9 punti)

In un sistema con memoria virtuale le pagine sono composte da 10000 (esadecimale) byte, la RAM è fatta di 2^{10} (decimale) frame, e lo spazio di indirizzamento logico massimo è di 4096 (decimale) pagine.

a) Qual è la lunghezza in bit di un indirizzo logico? 28 bit ($2^{16} \cdot 2^{12} = 2^{28}$)

Qual è la lunghezza in bit di un indirizzo fisico? 26 bit ($2^{16} \cdot 2^{10} = 2^{26}$)

b) Si consideri la PT sottostante (attenzione: nella tabella i numeri sono tutti in base decimale), e si dia l'indirizzo fisico in binario corrispondente ai seguenti indirizzi virtuali:

10AB75: =
n. di pag. = 10 (16 decimale), offset = AB75 → 3F AB75 = 00 0011 1111 1010 1011 0111 0101

113BC2
n. di pag.= 11 (17 decimale), bit di validità = "i" => page fault

n. pagina	n. frame	Valido/invalido
0	520	v
1	1001	v
2	x	i
3	x	i
4	x	v
5	75	v
6	1200	v
7	551	v
8	x	i
9	x	i
10	9	v
11	87	v
12	1824	v

13	1050	i
14	56	i
15	12	v
16	63	v
17	128	i

c) Nella tabella data, alcuni valori sono chiaramente sbagliati, dire quali e spiegare perché.

I numeri dei frame corrispondenti alle pagine 6, 12 e13 sono maggiori di 1024, il numero di frame in cui è suddivisa la memoria fisica del sistema.

d) Che cosa cerca di evitare un algoritmo di rimpiazzamento delle pagine?

Che una pagina selezionata come pagina vittima venga riferita nell'immediato futuro.
.

e) Dal punto di vista della memoria virtuale, perché il modo in cui i programmi accedono ai dati in RAM può pesantemente influire sulla velocità di esecuzione dei programmi stessi?

Vedere lucidi della sezione 9.9.5

ESERCIZIO 3 (9 punti)

a) In un sistema i blocchi su disco occupano 512 byte, e un puntatore a blocco è scritto su 4 byte. Il sistema operativo adotta l'allocazione indicizzata. Quanti accessi al disco sono necessari per leggere il byte numero 100.000, assumendo che gli attributi del file in questione siano già in memoria primaria? (motivate la vostra risposta, specificando le eventuali assunzioni che fate).

3. In un blocco indice possiamo scrivere 512/4 = 128 puntatori, indirizzando così 64Kbyte di dati, per cui un solo blocco indice non è sufficiente ad indirizzare il blocco contenente il byte specificato. Sia usando l'allocazione indicizzata gerarchica che l'allocazione indicizzata concatenata, un secondo blocco indice è sufficiente per indirizzare il blocco contenente il byte specificato. Sono quindi necessari due accessi al disco per leggere i due blocchi indice + un accesso per leggere il blocco contenente il byte numero 100.000.

b) Quali sono i vantaggi e gli svantaggi dell'allocazione indicizzata?

Vedere i lucidi della sezione 11.4.3

c) In un dato momento, lo spazio occupato sul disco da una cartella MYDIR è di X byte. Dopo un po' di tempo, lo spazio occupato dalla cartella MYDIR aumenta, ma la quantità totale di spazio occupato sul disco è diminuita. Come è possibile spiegare questa situazione? (si assuma per semplicità che vi sia un solo utente nel sistema, e che l'utente stia operando solo nella cartella MYDIR)

L'utente ha cancellato pochi file molto grossi da MYDIR, e ha inserito in MYDIR molti file piccoli la cui dimensione totale è inferiore alla dimensione dei file cancellati.

d) Cosa succede quando si esegue la *open* di un file, in un generico sistema operativo?

Si comunica al sistema che si vuole usare quel file: il SO recupera gli attributi del file e li copia in RAM, nella open file table. Successivi accessi agli attributi del file da parte del programma che ha chiamato la open useranno la copia degli attributi in RAM, anziché la copia su disco (ad accesso molto più lento). Di solito, anche una parte del file stesso viene copiata in RAM, in modo da velocizzare l'accesso ai dati del file. La open restituisce un *file pointer* che verrà usato dal programma per accedere ai vari dati/attributi del file disponibili in RAM.