UNIVERSITÀ DEGLI STUDI DI BERGAMO

Corso di laurea in ingegneria informatica

Esame di sistemi operativi – appello straordinario – 2 novembre 2006

Cognome_____Nome_____Matricola_____

1.

Si consideri il seguente programma:

```
main()
{  int pid, pid1, d;
   char c[70];
   ...
   pid=fork();
   if (pid == 0)
   {   /* prima parte di codice eseguito da Q figlio
di P */
       ….
       pid1=fork();
       if (pid1 == 0)
       {  /* codice eseguito da R figlio di Q */
           write(stdout, c, 70);
           exit(2);
       }else{   /* seconda parte di codice eseguito
da Q  */
            ……
            pid1 = wait(&status);
            exit(1);
       }
  }else
  {   /* codice eseguito dal padre P  */
      ……
      read(stdin, &d, 1);
      pid = wait(&status);
      exit(0);
  }
}
```

Un processo P esegue tale programma, creando a un certo momento un processo figlio chiamato Q che, a sua volta, crea un processo figlio chiamato R.

Nella tabella a pagina seguente sono indicati (nella prima colonna) alcuni eventi che si sono verificati durante l'esecuzione del programma da parte di P, Q e R; nella seconda colonna è aggiunta un'indicazione supplementare relativa a tali eventi. Si deve completare tale tabella indicando ordinatamente nella terza colonna tutti i moduli del S.O. che vengono eseguiti (completamente o in parte) in seguito all'evento, nella quarta colonna il contesto nel quale ciascun modulo è eseguito e, nelle ultime tre colonne, lo stato dei processi P, Q e R dopo che tutti i moduli hanno svolto la funzione e si ritorna al funzionamento in modo U.

Avvertenze per il riempimento della tabella:

1. indicare i moduli utilizzando la notazione seguente:

Notazione abbreviata	Modulo (o frammento di modulo) di sistema
G_SVC	Gestore SVC
R_Int(Disp)	Routine di interrupt; Disp può valere CK=orologio, Sout=Standard output, Sin=Standard input
<nome routine di sistema>	puo' essere: fork, write, read, wait, exit, preempt, change
Sleep_on(E_n)	Sleep_on. Per indicare l'evento su cui viene sospeso il processo usare convenzionalmente E_1, E_2, E_3, ...
Wakeup(E_n)	Wakeup. E_n indica l'evento, come in Sleep_on

2. non esistono altri processi nel sistema
3. dopo una fork viene ripresa l'esecuzione del processo padre; il processo figlio inizierà l'esecuzione all'interno del modulo fork
4. se c'è più di un processo in stato di pronto, viene mandato in esecuzione quello che da più tempo si trova in tale stato
5. il buffer del driver di standard output ha dimensione di 100 caratteri
6. la notazione 30 interrupt (40 interrupt) indica che si sono verificati 30 (40) interrupt. Nella risposta fare riferimento all'ultimo di tali interrupt
7. il modulo wait invoca sleep_on su un evento opportuno
8. la terminazione di un processo (exit) invoca wakeup per risvegliare il processo padre eventualmente in attesa.
9. se un processo viene risvegliato da wakeup e non ci sono altri processi pronti o in esecuzione, il processo viene immediatamente lanciato in esecuzione (in questo caso wakeup invoca change)

Evento (preceduto dal processo nel cui contesto l'evento si verifica)	**Informazioni aggiuntive**	**Moduli eseguiti per gestire l'evento**	**Processo/i nel cui contesto è eseguito ogni modulo**	**Stato dei processi dopo la gestione dell'evento**		
				P	**Q**	**R**
P: fork	P non ha ancora esaurito il suo quanto di tempo	G_SVC fork	P P	Esec U	pronto	non esiste
P: read	Lo standard input non è pronto	G_SVC read Sleep_on(E1) Change fork	P P P P-Q Q	Attesa	Esec U	non esiste
Q: fork	Q non ha esaurito il suo quanto di tempo	G_SVC fork	Q Q	Attesa	Esec U	pronto
Q: interrupt orologio	Q ha esaurito il suo quanto di tempo	R_int(CK) Preempt Change fork	Q Q Q-R R	Attesa	pronto	Esec U
R: interrupt da standard input	R non ha esaurito il suo quanto di tempo	R_int(Sin) Wakeup(E1)	R R	pronto	pronto	Esec U
R: write	write ha trasferito i 70 caratteri nel buffer	G_SVC write Sleep_on(E2) Change Preempt	R R R R-Q Q	pronto	Esec U	Attesa
Q: 30 interrupt da standard output	Q non ha esaurito il suo quanto di tempo	R_int(Sout)	Q	pronto	Esec U	Attesa
Q: wait		G_SVC wait Sleep_on(E3) Change Sleep_on(E1) read	Q Q Q Q-P P P	Esec U	Attesa	Attesa
P: 40 interrupt da standard output	l'ultimo è relativo all'ultimo carattere da trasferire; P non ha esaurito il suo tempo	R_int(Sout) Wakeup(E2)	P P	Esec U	Attesa	Pronto
P: wait		G_SVC wait Sleep_on(E4) Change Sleep_on(E2) write	P P P P-R R R	Attesa	Attesa	Esec U
R: exit		G_SVC exit Wakeup(E3) Change Sleep_on(E3) wait	R R R R-Q Q Q	Attesa	Esec U	non esiste

Q: exit		G_SVC exit Wakeup(E4) Change Sleep_on(E4) wait	Q Q Q Q-P P P	Esec U	non esiste	non esiste
P: exit		G_SVC exit	P P	non esiste	non esiste	non esiste

2.

Un sistema dotato di memoria virtuale con paginazione e segmentazione tipo UNIX è caratterizzato dai seguenti parametri: l'indirizzo logico è di 32 bit; l'indirizzo fisico è di 24 bit. La dimensione delle pagine è di 4K.

a. definire la struttura dell'indirizzo logico e di quello fisico indicando la lunghezza dei campi che li costituiscono:
NPV: 20 _____ Spiazzamento logico: 12_____

NPF: 12 _____ Spiazzamento fisico: 12 _____

b. nel sistema sono attivi 2 processi P e Q, che eseguono 2 programmi PGP e PGQ ed hanno un segmento dati condiviso. La dimensione iniziale dei segmenti dei due programmi è la seguente:
CP: 7K; DP: 11K; PP: 4K; COND: 6K;

CQ: 19K; DQ: 2K; PQ: 4K;

La dimensione complessiva di ognuno dei 2 processi è di 64K e quindi il segmento pila di ogni processo inizia all'indirizzo corrispondente ai 64K; in ambedue i processi il segmento COND è allocato lasciando 3 pagine libere dopo il segmento dati iniziale per permettere una crescita dello Heap (dati dinamici, servizio BRK).

Indicare, completando la seguente tabella, i numeri delle pagine virtuali assolute (NPV) che costituiscono i segmenti in ognuno dei due processi. Indicare gli NPV omettendo gli 0 iniziali e l'indicazione della notazione esadecimale (ad esempio, N invece di 0x0000N).

CP:	0, 1	CQ:	0, 1, 2, 3, 4 .....
DP:	2, 3, 4	DQ:	5.....
COND:	8, 9	COND:	9, A .....
PP:	F	PQ:	F.....

c. Indicare quanto spazio (in pagine) è disponibile per la crescita dei segmenti pila dei due processi
crescita di PP: 5_____ crescita di PQ: 4 _____

d. Ad un certo istante T sono avvenuti i seguenti eventi:
- lancio di P (fork di P ed exec di PGP)
- lancio di Q (fork di Q ed exec di PGQ)
- crescita della pila di P fino a 3 pagine

Sapendo che il numero di pagine residenti **R** è 10 e ipotizzando che l'allocazione delle pagine virtuali nelle pagine fisiche sia avvenuto in sequenza senza buchi a partire dalla pagina fisica E0 (anche nell'indicazione degli NPF sono omessi gli 0 inutili), indicare, completando la seguente tabella, l'allocazione fisica delle pagine dei due processi all'istante T.

E0:	CP0.....	E1:	CP1.....	E2:	DP0.....	E3:	DP1.....	E4:	DP2 .....
E5:	COND0.....	E6:	COND1.....	E7:	PP0	E8:	CQO.....	E9:	CQ1.....
EA:	CQ2	EB:	CQ3	EC:	CQ4.....	ED:	DQ0	EE:	PQ0 .....
EF:	PP1	F0:	PP2.....	F1:	.....	F2:	.....	F3:	.....

e. Indicare il contenuto della tabella delle pagine della MMU all'istante T completando la seguente tabella (usare P e Q come pid dei corrispondenti processi); ipotizzare che le righe della tabella siano state allocate ordinatamente man mano che venivano allocate le pagine di memoria virtuale

PID	NPV	NPF
P	0	E0
P	1	E1
P	2	E2
P	3	E3
P	4	E4
P	8	E5
P	9	E6
P	F	E7
Q	0	E8
Q	1	E9
Q	2	EA
Q	3	EB
Q	4	EC
Q	5	ED
Q	9	E5
Q	A	E6
Q	F	EE
P	E	EF
P	D	F0

f. A un istante T' successivo a T si è verificato il seguente evento: crescita del segmento dati di Q fino all'indirizzo limite 0x00007xxx (tramite invocazione del servizio BRK); sapendo che viene utilizzato l'algoritmo LRU e che le pagine meno utilizzate del processo Q sono le prime 3 pagine del codice (CQ0 meno utilizzata di CQ1 meno utilizzata di CQ2), indicare, completando la seguente tabella, in quali pagine fisiche sono allocate le pagine virtuali del segmento dati DQ

numero pagina nel segmento	NPV	NPF
DQ0	5	ED
DQ1	6	F1
DQ2	7	E8

3.

Illustrare in modo dettagliato e preciso il problema del pranzo dei filosofi e una sua possibile soluzione.

Fare riferimento al §7.6.3 del libro di testo.