UNIVERSITÀ DEGLI STUDI DI BERGAMO

# Esame di sistemi operativi – 13 febbraio 2007

Cognome_____Nome_____Matricola_____

**1.**

Un sistema dotato di memoria virtuale con paginazione e segmentazione tipo UNIX è caratterizzato dai seguenti parametri: l'indirizzo logico è di 17 bit; l'indirizzo fisico è di 17 bit. La dimensione delle pagine è di 8Kbyte.

(a) Definire la struttura dell'indirizzo logico e di quello fisico indicando la lunghezza dei campi che li costituiscono:

NPV: *4 bit*_____ Spiazzamento logico: *13 bit*_____

NPF: *4 bit*_____ Spiazzamento fisico: *13 bit* _____

(b) Nel sistema saranno attivati i processi P, Q e R, che eseguono i programmi PGP, PGQ e PGR e condividono un segmento dati. La dimensione iniziale dei segmenti dei tre programmi è la seguente:

CP: 24K; DP: 8K; PP: 8K; COND: 8K;

CQ: 16K; DQ: 8K; PQ: 16K; COND: 8K;

CR: 16K; DR: 8K; PR: 8K; COND: 8K;

La dimensione complessiva di ognuno dei 3 processi è di 128K e quindi il segmento pila di ogni processo inizia all'indirizzo corrispondente ai 128K; nel processo P il segmento COND è allocato lasciando 3 pagine libere dopo il segmento dati (per permettere una crescita dello Heap, servizio BRK) nei processi Q e R COND è allocato lasciando 2 pagine libera dopo DQ e DR rispettivamente.Inserire in tabella 1a il significato delle varie pagine di memoria logica (notazione: CP0, CP1, DP0, PP0,...CQ0, ..., COND0,...).

| **Indirizzo di pagina virtuale** | **Processo P** | **Processo Q** | **Processo R** |
|---|---|---|---|
| 0 | *CP0* | *CQ0* | *CR0* |
| 1 | *CP1* | *CQ1* | *CR1* |
| 2 | *CP2* | *DQ0* | *DR0* |
| 3 | *DP0* | | |
| 4 | *DP1* | | |
| 5 | *DP2* | *COND0* | *COND0* |
| 6 | | | |
| 7 | *COND0* | | |
| 8 | | | |
| 9 | | | |
| A | | | |
| B | | | |
| C | | | |
| D | | *PQ2* | *PR2* |
| E | | *PQ1* | *PR1* |
| F | *PP0* | *PQ0* | *PR0* |

1A) Struttura della Memoria Logica

| **Indirizzo fisico** | **Pagine allocate $t_0$** | **Indirizzo fisico** | **Pagine allocate $t_1$** |
|---|---|---|---|
| 0 | **Occupata** | 0 | *PR2* |
| 1 | **Occupata** | 1 | *CR1* |
| 2 | *CQ0* | 2 | *PQ2* |
| 3 | *CQ1* | 3 | *CQ1* |
| 4 | *DQ0* | 4 | *DQ0* |
| 5 | *COND* | 5 | *COND* |
| 6 | *PQ0* | 6 | *PQ0* |
| 7 | *PQ1* | 7 | *PQ1* |
| 8 | *CP0* | 8 | *DP1* |
| 9 | *CP1* | 9 | *DP2* |
| A | *CP2* | A | *CP2* |
| B | *DP0* | B | *DP0* |
| C | *PP0* | C | *PP0* |
| D | | D | *DR0* |
| E | | E | *PR0* |
| F | | F | *PR1* |

1B) Memoria Fisica agli istanti $t_0$ e $t_1$

(c) Indicare quanto spazio (in pagine) è disponibile per la crescita dello Heap nel processo Q e dello stack nel processo R

Q: *2* _____ R: *9* _____

(d) Ad un certo istante $t_0$ sono terminati i seguenti eventi:

1. allocazione delle pagina fisiche di indirizzo 0 e 1 a un processo diverso da P,Q,R
2. lancio di Q (fork di Q ed exec di PGQ)
3. lancio di P (fork di P ed exec di PGP)

Sapendo che il numero di pagine residenti **R** è 6, che viene utilizzato l'algoritmo LRU e che le pagine meno utilizzate in ogni processo sono la prima e la seconda pagina del segmento codice (le prime caricate sono le meno utilizzate) e ipotizzando che l'allocazione delle pagine virtuali nelle pagine fisiche avvenga in sequenza senza buchi a partire dalla pagina fisica 0, indicare, completando tabella 1B (sinistra), l'allocazione fisica delle pagine dei tre processi all'istante $t_0$. Si indica con **occupata** una pagina utilizzata da altro processo.

(e) Ad un certo istante $t_1 > t_0$ sono terminati i seguenti eventi:

4. crescita della pila di Q di 1 pagina
5. allocazione di 2 pagina di memoria per P (servizio BRK)
6. deallocazione delle due pagine occupate da altro processo
7. lancio di R (fork di R ed exec di PGR)
8. crescita della pila di R di 2 pagine

Completare la tabella 1B (destra) nelle medesime ipotesi delineate nel precedente punto (d) e assumendo che, dovendo utilizzare una pagina fisica libera, venga sempre utilizzata la pagina fisica libera avente indirizzo di pagina minore.

(f) Indicare il contenuto della tabella delle pagine della MMU all'istante $t_1$ assumendo una configurazione vergine per la MMU e completando la seguente tabella (usare P, Q e R come pid dei corrispondenti processi, oppure NS se nella corrispondente riga della tabella non è allocato alcun processo e quindi è non significativa, **-** se la pagina è occupata da un processo diverso da P,Q e R)); ipotizzare che le righe della tabella siano state allocate ordinatamente man mano che venivano allocate le pagine di memoria virtuale e che gli eventi di cui al punto (e) influenzanti la MMU abbiano utilizzato le righe lasciate libere e che **se richiesta una nuova riga si utilizzi la prima riga libera**. Indicare anche il valore assunto dal bit di validità di pagina (1 significa che la pagina è caricata).

| PID | Num. Pag. Virt. | Num. Pag. Fis. | Bit Validità |
|---|---|---|---|
| R | *D* | *0* | *1* |
| R | *1* | *1* | *1* |
| Q | *D* | *2* | *1* |
| Q | *1* | *3* | *1* |
| Q | *2* | *4* | *1* |
| Q | *5* | *5* | *1* |
| Q | *F* | *6* | *1* |
| Q | *E* | *7* | *1* |
| P | *4* | *8* | *1* |
| P | *5* | *9* | *1* |
| P | *2* | *A* | *1* |
| P | *3* | *6* | *1* |
| P | *7* | *5* | *1* |
| P | *F* | *C* | *1* |
| R | *2* | *D* | *1* |
| R | *5* | *5* | *1* |
| R | *F* | *E* | *1* |
| R | *E* | *F* | *1* |
| NS | *-* | *-* | *0* |

**2.**
Illustrare, anche con l'aiuto di un diagramma appropriato, tutti gli stati in cui si può trovare un processo e le cause che comportano il passaggio da uno stato all'altro.

Confrontare il §4.1 del libro di testo

**3.**

Un processo P esegue il seguente programma, creando un processo figlio Q, che crea a sua volta un figlio R:

```
int main ( ) { /* processo P */
   /* dichiarazioni varie */
   fd1 = open ("/cat1/cat3/file3", O_RDONLY);
   read (fd1, bufP1, 5);
   fd2 = open ("/cat1/cat2/file1", O_RDONLY);
   read (fd2, bufP2, 4);
   pid = fork ( );
   if (pid == 0) { /* processo Q */
      lseek (fd2, -4, 2); //2 = rif. relativo alla fine del file
      read (fd2, bufQ, 3);
      lseek (fd2, -8, 1); //1 = rif. relativo alla pos. cor.
      fd3 = open ("/cat1/cat2/file2", O_RDONLY);
      pid = fork ( );
      if (pid == 0) { /* processo R */
         read (fd2, bufR2, 4);
         read (fd3, bufR3, 5);
         close (fd2);
         exit (0);
      } /* fine R */
   } /* fine Q */
} /* fine P */
```

A un certo istante T si sono verificati gli eventi seguenti: 1) dopo l'esecuzione della prima **fork** è andato in esecuzione subito il processo Q; 2) dopo l'esecuzione della seconda **fork** è andato in esecuzione subito il processo R; 3) il processo R è arrivato all'invocazione della funzione **exit** (che non è ancora stata eseguita).
Il contenuto del volume durante l'esecuzione è il seguente:

**I-Lista:** <0,dir,5> <2,dir,21> <13,dir,16> <9,dir,11> <55,norm,85> <57,norm,70> <85,norm,55>
**Blocco 5:** ... < 1, dev >, < 2, cat1> ...
**Blocco 11:** ... < 55, file1> ... <57, file2> ...
**Blocco 16:** ... < 85, file3>
**Blocco 21:** ... < 13, cat3 > ... <9, cat2> ...
**Blocco 55:** Meccanica
**Blocco 70:** Matematica
**Blocco 85:** Architettura

1) Indicare il contenuto, all'istante T, delle variabili seguenti:

bufP1: ***Mecca*** _____ bufP2: ***Arch*** _____ bufQ: ***tur*** _____

bufR2: ***hite*** _____ bufR3: ***Matem*** _____

2) Completare il contenuto delle tabelle seguenti, all'istante T:

| Tab. File Aperti Proc. P | | Tab. File Aperti Proc. Q | | Tab. File Aperti Proc. R | | Tabella Globale dei File Aperti | | | |
|---|---|---|---|---|---|---|---|---|---|
| **f.d.** | **rif.** | **f.d.** | **rif.** | **f.d.** | **rif.** | **riga** | **pos. corr.** | **n. di proc.** | **punt.** |
| 0 | xxx | 0 | xxx | 0 | xxx | 0 | xxx | xxx | xxx |
| 1 | xxx | 1 | xxx | 1 | xxx | 1 | xxx | xxx | xxx |
| 2 | xxx | 2 | xxx | 2 | xxx | 2 | xxx | xxx | xxx |
| 3 | ***3*** | 3 | ***3*** | 3 | ***3*** | 3 | ***5*** | ***3*** | ***85*** |
| 4 | ***4*** | 4 | ***4*** | 4 | ***L*** | 4 | ***7*** | ***2*** | ***55*** |
| 5 | | 5 | ***5*** | 5 | ***5*** | 5 | ***5*** | ***2*** | ***57*** |

***pos. corr.*** *è l'indicatore di posizione corrente all'interno del file;* ***punt.*** *è lo i-number che fa riferimento allo i-node;* ***n. di proc.*** *è il numero di processi che hanno aperto il file;* ***xxx*** *indica che è presente un valore non significativo ai fini del problema; le tabelle sono semplificate e contengono solamente le colonne delle quali è richiesto il riempimento; si scriva* ***L*** *per indicare che una cella precedentemente utilizzata è diventata libera.*