Sistemi Operativi L-A
Prova Scritta di Recupero del 5 Aprile 2004

1. Unix [12 punti]

Si scriva un programma C che utilizzi le system call di UNIX e che realizzi un comando UNIX avente la seguente sintassi:

```
esame fsize fin fout dmax fmax
```

dove:

- `fsize` è un file che contiene una sequenza di interi (2 byte). In particolare,ciascun intero indica la dimensione di un blocco di byte;
- `fin` è un file di testo;
- `fout` è il nome di un file;
- `dmax`, `fmax` sono due interi.

Il comando deve funzionare nel modo seguente:

1. Il processo iniziale, P0, deve creare due processi figli: P1 e P2.

2. All'avvio, P1 comincia a leggere da `fsize` e da fin: in particolare, P1 deve leggere da `fsize` la dimensione *d* del prossimo blocco di byte da leggere da `fin`, e conseguentemente comunicherà a P1 (i) l'intero *d*, e (ii) il blocco di *d* byte letti da `fin`.

3. P2, nel caso in cui d sia maggiore di `dmax`, deve incrementare un contatore *c* (inizializzato a zero). Quando il contatore *c* arriva al valore 3, deve: (a) reinizializzare il contatore *c*, e (b) avvisare P1 che il blocco di *d* byte deve essere scritto su `fout`. In tutti gli altri casi (se *d* è minore o uguale a `dmax` o se *c* è diverso da 3), P2 stampa su video i *d* byte ricevuti da P1.

4. Se P1 viene 'avvisato' da P2, deve fare in modo che il blocco di *d* byte in oggetto venga scritto su `fout`.

5. Dopo `fmax` byte scritti su `fout`, l'applicazione deve terminare (non devono rimanere processi attivi o zombie).

Per semplicità, si assuma:

- che i file abbiano i diritti necessari (non è richiesto gestire errori sulla creazione/lettura/scrittura relative a file);
- che il comando sia invocato correttamente (non è necessario effettuare il controllo sui parametri);
- che gli interi in `fsize` siano tutti minori di 100.

2. Teoria. [7 punti]

La comunicazione tra processi nel sistema UNIX.

Sistemi Operativi L-A
Prova Scritta di Recupero del 5 Aprile 2004

3. Teoria. [7 punti]

Scheduling della CPU.

4. Sincronizzazione tra Processi: [6 punti]

```
#define MAX1 2
#define MAX2 6
shared semaphore s1=0,s2=3;
shared int V=0;
```

```
/* Processo P1 */
main()
{
  int cont=MAX1;
     while(cont)
    {
        wait (&s1);
        V++;
        printf("T1:%d\n", V);
        if (V==MAX1+MAX2)
                signal(&s2);
        signal(&s1);
        cont--;
    }
}
```

```
/* Processo P2 */
main()
{
     int cont=MAX2;
        while(cont)
        {
        wait (&s2);
        V+=2;
        printf("T2:%d%d\n",cont,V);
        if (V==MAX2)
        {  cont=0;
            signal(&s1);

        }
    }
}
```

Sistemi Operativi L-A
Prova Scritta di Recupero del 14 Gennaio 2004

1. Unix [12 punti]

Si scriva un programma C che utilizzi le system call di UNIX e che realizzi un comando UNIX avente la seguente sintassi:

```
esame filein
```

dove:

- **esame** è il nome del comando;
- **filein** è il nome di un file accessibile in lettura;

In particolare, `filein` è un file di testo, composto di righe di 6 caratteri l'uno, ciascuna contenente

- il nome di un comando eseguibile, oppure
- la stringa "`report`"

Il comando deve funzionare nel modo seguente:

- inizialmente, il processo iniziale, **P0**, crea un processo figlio: **P1**;
- successivamente P0 deve leggere una riga alla volta da `filein`; per ogni riga letta:
  - **se la riga coincide con la stringa `'report'`**, il padre deve comunicare a P1 il numero di righe finora lette; al ricevimento di tale messaggio, P1 stamperà a video il valore ricevuto da P0;
  - **in caso contrario** (cioe`, la riga letta non e` la stringa `'report'`, ma una stringa diversa che rappresenta il nome di un comando C), P0 deve creare un nuovo figlio, il quale deve eseguire il comando C;
- Dopo la creazione di ogni nuovo figlio, **P0** deve continuare la propria esecuzione, leggendo una nuova riga da `filein` (è possibile, quindi, che più processi figli eseguano comandi in parallelo, ognuno relativo una diversa riga di filein), oppure terminare la propria esecuzione (se la lettura di filein e` terminata).

Il processo P0, prima di terminare, deve provocare la terminazione di P1.

2. Teoria. [7 punti]

Il processo Unix: caratteristiche e rappresentazione.

3. Teoria. [7 punti]

Memoria virtuale.

Sistemi Operativi L-A
Prova Scritta di Recupero del 14 Gennaio 2004

4. Sincronizzazione tra Processi: [6 punti]

Analizzare la seguente applicazione concorrente (scritta in pseudo-C), composta da 2 processi, nell'ipotesi di **modello ad ambiente globale**. Commentare il funzionamento dell'applicazione e mostrare l'eventuale output prodotto.

```
#define K1 4
#define K2 4
shared semaphore s1=0,s2=1;
shared int X=10;
```

```
/* Processo P1 */
main()
{   int cont=K1;
     while(cont)
    {
        wait (&s1);
        X--;
        printf("P1: X=%d\n", X);
        if (X<0){
                cont-=K1;
                signal(&s2);
        }
        else signal(&s1);
    }

}
```

```
/* Processo P2 */
main()
{   int cont=K2;
     while(cont)
    {
        wait (&s2);
        X-=cont;
        printf("P2: (X=%d)\n", X);

        if (X<K1)
        {       cont=0;
                signal(&s1);
        }
        else signal(&s2);
    }

}
```

Sistemi Operativi L-A
Prova Scritta di Recupero del 15 Settembre 2003

COMPITO A

1. Unix [12 punti]

Si scriva un programma C che utilizzi le system call di UNIX e che realizzi un comando UNIX avente la seguente sintassi:

```
esame filein n m c
```

dove:

- **esame** è il nome del comando;
- **fileout** è il nome di un file accessibile in scrittura;
- **n** ed **m** sono numeri interi positivi;
- **c** è un carattere.

Il comando deve funzionare nel modo seguente:

- il processo 'padre', P0, crea tre processi figli: P1, P2 e P3,quindi si pone in attesa della loro terminazione; una volta che P1, P2 e P3 sono terminati, P0 deve chiudere il file fileout e a sua volta terminare.
- il processo P1 deve leggere da standard input: per ogni n caratteri letti comunica il carattere n-esimo al processo P2;
- il processo P2, ogni volta che riceve un carattere da P1, lo confronta con c. Se è diverso da c, lo stampa a video. Se invece riceve da P1 il carattere c, invia al padre un segnale (in questo caso, il padre reagisce al segnale stampando un carattere '*' su fileout);
- il processo P3 implementa un timer: non appena viene creato si mette in attesa per m secondi. Allo scadere degli m secondi deve terminare i processi P1 e P2, e quindi deve esso stesso terminare.

2. Teoria. [7 punti]

Sincronizzazione tra processi nel sistema operativo Unix..

3. Teoria. [7 punti]

Il file system.

Sistemi Operativi L-A
Prova Scritta di Recupero del 15 Settembre 2003

4. Sincronizzazione tra Processi: [6 punti]

Analizzare la seguente applicazione concorrente (scritta in pseudo-C), composta da 2 processi, nell'ipotesi di **modello ad ambiente globale**. Commentare il funzionamento dell'applicazione e mostrare l'eventuale output prodotto.

```
#define N1 4
#define N2 4
shared semaphore s1=N1,s2=0;
shared int X=N1+N2;
shared int cont=1;
```

```
/* Processo P1 */
main()
{
     while(cont)
    {
        wait (&s1);
        X--;
        printf("P1: X=%d\n", X);
        if (X<=N1){
                cont++;
                signal(&s2);
        }
    }

}
```

```
/* Processo P2 */
main()
{
     while(cont)
    {
        wait (&s2);
        X-=cont;
        printf("P2: (X=%d)\n", X);

        if (X<0)
        {       cont=0;
                signal(&s1);
        }
        else signal(&s2);
    }

}
```

Sistemi Operativi L-A
Prova Scritta di Recupero del 15 Settembre 2003

COMPITO B

1. Unix [12 punti]

Si scriva un programma C che utilizzi le system call di UNIX e che realizzi un comando UNIX avente la seguente sintassi:

```
esame filein n m c
```

dove:

- **esame** è il nome del comando;
- **fileout** è il nome di un file accessibile in scrittura;
- **n** ed **m** sono numeri intero positivo;
- **c** è un carattere.

Il comando deve funzionare nel modo seguente:

- il processo 'padre', P0, crea tre processi figli: P1, P2 e P3, quindi si pone in attesa della loro terminazione;
- il processo P1 deve leggere da standard input: per ogni n caratteri letti comunica i primi n-1 caratteri al processo P2, stampa invece l'n-esimo carattere su fileout ;
- il processo P2, ogni volta che riceve un carattere da P1, lo confronta con c. Se è diverso da c, lo stampa a video. Se invece riceve da P1 il carattere c, invia al padre un segnale (in questo caso, il padre reagisce al segnale incrementando un contatore);
- il processo P3 implementa un timer: non appena viene creato si mette in attesa per m secondi. Allo scadere degli m secondi deve avvisare il padre;
- appena il padre riceve il segnale di terminazione da P3, deve provocare la terminazione dei tre figli: una volta che i figli sono terminati, deve stampare a video il valore del contatore.

2. Teoria. [7 punti]

Comunicazione tra processi nel sistema operativo UNIX.

3. Teoria. [7 punti]

La memoria virtuale.

Sistemi Operativi L-A
Prova Scritta di Recupero del 15 Settembre 2003

4. Sincronizzazione tra Processi: [6 punti]

Analizzare la seguente applicazione concorrente (scritta in pseudo-C), composta da 2 processi, nell'ipotesi di **modello ad ambiente globale**. Commentare il funzionamento dell'applicazione e mostrare l'eventuale output prodotto.

```
#define N1 0
#define N2 1
shared semaphore s1=N1,s2=N2;
shared int X=4;
shared int cont=1;
```

```
/* Processo P1 */
main()
{
     while(cont)
    {
        wait (&s1);
        X+=3;
        printf("P1: X=%d\n", X);
        if ((X%2)==0)
              signal(&s2);
        else
              signal(&s1);

    }

}
```

```
/* Processo P2 */
main()
{
     while(cont)
    {
        wait (&s2);
        X++;
        printf("P2: (X=%d)\n", X);

        if ((X%2)==1)
        {    if (X==15)cont=0;
            signal(&s1);
        }
        else signal(&s2);
    }

}
```

SISTEMI OPERATIVI L-A
Prova scritta parziale finale del 18 Marzo 2003

COMPITO A

Domanda 1 [13 punti]

Si scriva un programma in C che realizzi un comando che, utilizzando le system call di unix, preveda la seguente sintassi:

```
esame filein car n1 n2
```

dove:

- **`filein`** e` un file su cui si ha accesso in lettura,
- **car** e` un singolo carattere,
- **`n1,n2`** sono due interi positivi.

Il programma dovra` funzionare nel modo seguente:

- il processo iniziale P0 deve creare due processi figli (P1 e P2);
- il figlio P1, dopo n1 secondi (dalla sua creazione) manda un segnale a P0;
- il figlio P2, dopo n2 secondi (dalla sua creazione) manda un segnale a P0;
- quando P0 ha ricevuto **entrambi i segnali**, comincia a leggere da **`filein`** e comunica i caratteri letti dal file ai figli mediante un canale di comunicazione;
- i figli, in parallelo, prelevano dal canale (uno alla volta) i caratteri spediti dal padre; si noti che, riguardo all'accesso dei figli al canale, non e` stabilito alcun vincolo e non e` pertanto possibile prevedere l'ordine di accesso dei figli al canale;
- al prelievo di un carattere C, ogni figlio Pi (i=1,2) lo confronta con il carattere **`car`** dato come argomento:
  - se **C e` uguale a `car`,** il figlio Pi provoca la terminazione di tutti i processi;
  - altrimenti (se **C e` diverso da `car`**) Pi stampa il carattere C su standard output.
- se il carattere **`car`** non viene mai incontrato, una volta raggiunta la fine del file **`filein`** il padre aspetta la terminazione dei figli e poi termina.

Domanda 2 [10 punti]

Memoria virtuale con paginazione su domanda.

SISTEMI OPERATIVI L-A
Prova scritta parziale finale del 18 Marzo 2003

Domanda 3 [8 punti]

Analizzare la seguente applicazione concorrente (scritta in pseudo-C), composta da 2 processi, nell'ipotesi di **modello ad ambiente globale**. Commentare il funzionamento dell'applicazione e mostrare l'eventuale output prodotto.

```
#define MAX1 4
#define MAX2 5
shared semaphore s1=0,s2=3;
shared int V=0;
```

```
/* Processo P1 */
main()
{
  int cont=MAX1;
     while(cont)
    {
        wait (&s1);
        V++;
        printf("P1:%d\n", V);
        if (V==2*MAX1)
                signal(&s2);
        signal(&s1);
        cont--;
    }
}
```

```
/* Processo P2 */
main()
{
     int cont=MAX2;
        while(cont)
        {
        wait (&s2);
        V+=2;
        printf("P2:%d%d\n",cont,V);
        if (V==MAX2)
        {   cont=0;
            signal(&s1);

        }
    }
}
```

SISTEMI OPERATIVI L-A
Prova scritta parziale finale del 18 Marzo 2003

COMPITO B

Domanda 1 [13 punti]

Si scriva un programma in C che realizzi un comando che, utilizzando le system call di unix, preveda la seguente sintassi:

```
esame filein c1 c2 n
```

dove:

- **`filein`** e` un file su cui si ha accesso in lettura,
- **c1, c2** sono singoli caratteri,
- **`n`** e` un intero positivo.

Il programma dovra` funzionare nel modo seguente:

- il processo iniziale P0 deve creare due processi figli (P1 e P2); dopo **`n`** secondi P0 "attiva" i due figli mediante la notifica asincrona di un evento;
- una volta attivati, entrambi i figli P0 e P1 leggono da **`filein`**, un carattere alla volta (si faccia in modo che ciascun carattere del file **`filein`** sia letto da un solo processo):
- ogni figlio Pi (i=1,2) confronta ogni carattere C letto con **`ci`** (i=1,2):
  - **se C e` diverso da `ci`**, il processo Pi comunica il carattere C a P0;
  - **se C e` uguale a `ci`,** Pi deve provocare la terminazione del padre e quindi della famiglia di processi (si faccia in modo che il padre attenda la terminazione dei figli).

Domanda 2 [10 punti]

Accesso a file di processi unix: strutture dati del sistema operativo e loro gestione.

SISTEMI OPERATIVI L-A
Prova scritta parziale finale del 18 Marzo 2003

Domanda 3 [8 punti]

Analizzare la seguente applicazione concorrente (scritta in pseudo-C), composta da 2 processi, nell'ipotesi di **modello ad ambiente globale**. Commentare il funzionamento dell'applicazione e mostrare l'eventuale output prodotto.

```
#define MAX1 3
#define MAX2 3
shared semaphore s1=1,s2=0;
shared int V=0;
shared int cont=1;
```

```
/* Processo P1 */
main()
{
      while(cont)
   {
       wait (&s1);
       V++;
       printf("P1:%d\n", V);
       if (V<2*MAX1)
              signal(&s2);
       else
       {  cont--;
     signal(&s2);
       }
   }
```

```
/* Processo P2 */
main()
{
        while(cont)
        {
        wait (&s2);
        if (cont)
        {V+=2;
         printf("P2 :%d\n",V);
        }
        if (V<MAX1*MAX2)
          signal(&s1);
        else cont=0;

        }
  }
```