Prova scritta di Sistemi di Elaborazione
Ingegneria delle Telecomunicazioni

Ing. G. Lettieri, Ing. A. Vecchio

30 giugno 2008

1. Supponiamo di avere il seguente programma scritto in parte in Assembler e in parte in C++:

```
.text
.global f2
f2:     pushl %ebp
        movl  %esp, %ebp
        pushl %ebx
        pushl %ecx
        pushl %edx
        pushl %esi
        movl  8(%ebp), %ebx
        movl  $0, %esi
        movl  $0, %ecx
l1:     cmpl  N, %esi
        jge   l2
        movl  %esi, %eax
        imull  N
        addl  12(%ebp), %eax
        addl  (%ebx, %eax, 4), %ecx
        incl  %esi
        jmp   l1
l2:     movl  %ecx, %eax
        popl  %esi
        popl  %edx
        popl  %ecx
        popl  %ebx
        leave
        ret
```

```
#include <stdio.h>
const int N = 3;
int f2(int a[N][N], int c);
int f1(int a[N][N])
{
        int b;
        int x;
        int y;
        b = f2(a, 0);
        for (x = 1; x < N; x++) {
                y = f2(a, x);
                if (y > b)
                        b = y;
        }
        return b;
}

int main()
{
        int i, j, r;
        int m[N][N];
        for (i = 0; i < N; i++)
                for (j = 0; j < N; j++)
                        scanf("%d", &m[i][j]);
        r = f1(m);
        printf("%d\n", r);
        return 0;
}
```

(a) Dire cosa viene calcolato dal programma complessivo.

(b) Tradurre la funzione `f1` in Assembler.

2. Scrivere i seguenti programmi in C++, utilizzando le primitive di Unix e la libreria standard del C.

   (a) Un programma `corridore` con argomenti da riga di comando *num* e *car*, dove *num* deve essere un numero intero maggiore o uguale a 0, mentre *car* deve essere un singolo carattere compreso tra 'a' e 'z' (inclusi). Il programma deve leggere il file numero *num* nella directory corrente (un carattere alla volta, usando la primitiva `read`), stampando un *car* sull'uscita standard per ogni carattere letto. Arrivato alla fine del file, deve stampare *car* in maiuscolo e terminare. Niente altro deve essere inviato sull'uscita standard. Per sapere qual è il file numero *num* nella directory corrente, il programma deve aprirla e contare i file nell'ordine restituito da `readdir`, partendo da 0 e saltando eventuali directory.

   (b) Un programma `gara` con un argomento *quanti* da riga di comando, dove *quanti* è un numero maggiore o uguale a 0 (ed eventualmente minore di un massimo prestabilito). Il programma deve creare *quanti* processi figli, in modo che il figlio i-esimo esegua il programma `corridore` con primo argomento pari a i, e secondo argomento pari alla i-esima lettera dell'alfabeto. Al termine, il programma `gara` deve mostrare una uscita del genere (esempio con $quanti = 5$):

   ```
   Vincitore: 3
   Tabella:
   0: 395
   1: 317
   2: 36
   3: 1536
   4: 1
   ```

   Nell'uscita, il "vincitore" è il primo `corridore` che ha stampato un carattere maiuscolo, mentre la "tabella" mostra quanti caratteri erano stati stampati dal `corridore` i-esimo fino a quel momento. Il programma mostra l'uscita e termina non appena viene decretato un vincitore, quindi attende che tutti i processi figli siano terminati prima di terminare esso stesso. (**Suggerimenti**: (i) fare in modo che tutti i processi figli inviino la propria uscita su un'*unica* pipe, letta un carattere alla volta dal processo padre; (ii) per costringere i figli a terminare, chiudere la pipe).