Prova scritta di Sistemi di Elaborazione
Ingegneria delle Telecomunicazioni

Ing. G. Lettieri, Ing. A. Vecchio

15 Gennaio 2007

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

```
.text
.global f1
f1:     pushl %ebp
        movl  %esp, %ebp
        pushl %esi
        movl  $0, %esi
l1:     cmpl  12(%ebp), %esi
        jae   fine
        pushl %esi
        pushl 12(%ebp)
        pushl 8(%ebp)
        call  f2
        addl  $12, %esp
        incl  %esi
        jmp   l1
fine:   popl  %esi
        leave
        ret
```

```
#include <stdio.h>
const int MAXN = 1000;

void f1(int v[], int n);

void f2(int v[], int n, int i)
{
        int j, tmp;
        int index = i;
        for(j=i; j<n; j++)
                if(v[j]<v[index])
                        index = j;
        tmp = v[index];
        v[index] = v[i];
        v[i] = tmp;
}
```

```
int main(int argc, char* argv[])
{
        int vett[MAXN];
        int n = 0;
        int i;
        FILE* ff = fopen(argv[1], "r");
        while(fscanf(ff, "%d", &i) == 1)
                vett[n++] = i;
        fclose(ff);
        f1(vett, n);
        for(i=0; i<n; i++)
                printf("%d\n", vett[i]);
        return 0;
}
```

   (a) Dire cosa viene calcolato dal programma complessivo

   (b) Tradurre la funzione `f2` in Assembler.

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

   (a) Un programma `tick`, con un argomento (da riga di comando) *max*. L'argomento deve essere un numero intero maggiore di zero. Il programma deve scegliere un numero casuale compreso tra 1 e *max* e stamparlo, una volta al secondo, sulla sua uscita standard (ogni volta su una nuova linea ed eseguendo una `fflush`). Se il programma riceve una signal `SIGUSR1`, deve scegliere un nuovo numero casuale da stampare, sempre compreso tra 1 e `max`. Se, invece, il programma riceve una signal `SIGUSR2`, deve terminare.

   (b) Un programma `listen`, con un due argomenti (da riga di comando) *target* e *max*. Entrambi gli argomenti devono essere numeri interi maggiori di zero. Il programma deve creare due processi, collegati tramite una pipe, in modo che il primo processo esegua il programma `tick` con argomento *max* e il secondo processo legga i numeri stampati dal primo. Il secondo processo deve continuare a leggere fino a quando non riceve un numero maggiore o uguale a *target*. Quando ciò avviene, si deve fare in modo che sia il primo che il secondo processo terminino. Il secondo processo, ogni volta che legge un numero minore di *target*, deve inviare un segnale al primo processo in modo da fargli generare un nuovo numero casuale.