Prova scritta di Sistemi di Elaborazione
Ingegneria delle Telecomunicazioni

Ing. G. Lettieri, Ing. A. Vecchio

20 Luglio 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 %ebx
        pushl %ecx
        movl  8(%ebp), %ebx
        movb  (%ebx), %cl
        cmpb  $'a, %cl
        jl    l1
        cmpb  $'z, %cl
        jg    l1
        subb  $'a, %cl
        movl  $0, %eax
        movb  %cl, %al
        jmp   l2
l1:     movl  $-1, %eax
l2:     popl  %ecx
        popl  %ebx
        leave
        ret
```

```
#include <stdio.h>
void f2(char *p, int vv[])
{
        int x = f1(p);
        if (x != -1)
                vv[x]++;
}
void f4(int n)
{
        printf("%d\n", n);
}
void f3(int ww[])
{
        int i;
        for (i = 0; i < 26; i++)
                f4(ww[i]);
}
int main(int argc, char *argv[])
{
        int i = 1;
        int zz[26] = { 0 };
        while (argv[i] != NULL) {
                f2(argv[i], zz);
                i++;
        }
        f3(zz);
        return 0;
}
```

(a) Dire cosa viene calcolato dal programma complessivo.

(b) Tradurre le funzioni `f2` e `f3` in Assembler.

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

   (a) Un programma `occupa` con argomenti *dir*, *nome* e *tempo* da riga di comando. Si suppone che la directory *dir* (che deve esistere già) contenga dei file il cui nome è un numero compreso tra 0 e un massimo stabilito a-priori. Ogni file rappresenta un posto occupato (per esempio in un teatro). Il programma deve trovare un posto libero, occuparlo per conto di *nome* per *tempo* secondi, quindi liberarlo. Per far ciò, il programma deve trovare un numero n non contenuto nella directory *dir*, creare un file di nome n e scriverci dentro *nome*, sospendersi per *tempo* secondi e, infine, eliminare il file n. Se tutti i posti sono occupati, il programma deve scrivere "*nome* ND" sullo standard error ed uscire. **Suggerimento**: quando si passano i flag `O_CREAT` e `O_EXCL` alla funzione `open`, questa restituisce -1 e pone il valore `EEXIST` nella variabile `errno` se il file da creare esisteva già.

   (b) Un programma `simula` che riceve da riga di comando un primo argomento *dir* e zero o più argomenti $nome_1$, $nome_2$, ... Il programma deve creare un processo per ogni parametro $nome_i$ ricevuto. Il processo figlio i-esimo deve eseguire il programma `occupa` con parametri *dir*, $nome_i$ e un *tempo* scelto casualmente tra 0 e 10. Dopo aver creato ciascun figlio, il processo padre deve sospendersi per 1 secondo. Lo standard error di tutti i figli deve essere rediretto, in modalità append, sul file "log.txt" che, se non esiste, va creato. Infine, il programma deve attendere la terminazione di tutti i suoi figli e terminare esso stesso.