Soluzioni della Prova Scritta di Sistemi di Elaborazione Ingegneria delle Telecomunicazioni

Ing. G. Lettieri, Ing. A. Vecchio

15 Gennaio 2007

1. (a) La funzione `f2` riceve come parametri l'indirizzo di partenza di un vettore `vett` di interi, il numero `n` di elementi conenuti nel vettore, e un indice `i`. La funzione scorre gli elementi di `vett` relativi alle posizioni con indici compresi tra `i` (incluso) e `n` (escluso), trova il valore più piccolo e lo scambia con quello in posizione `i`-esima. La funzione `f1` chiama `n` volte la funzione `f2`, passandole ogni volta il vettore `vett`, la sua dimensione, e un valore di `i` che va da 0 (incluso) a `n` (escluso). Al termine della chiamata ad `f1` il vettore `vett` è ordinato in modo crescente.

   La funzione `main` inserisce nel vettore `vett` gli interi contenuti nel file il cui nome viene specificato da riga di comando, chiama `f1` e stampa a video il valore degli elementi di `vett` che adesso sono ordinati in modo crescente.

   (b) Una possibile traduzione è la seguente:

```
        .text
        .global f2
f2:     pushl   %ebp
        movl    %esp, %ebp
        pushl %eax  # v
        pushl   %ebx  # tmp e altro
        pushl %ecx  # index
        pushl   %edx  # j
        pushl   %esi
        pushl %edi
        movl    16(%ebp), %ecx
        movl    16(%ebp), %edx
        movl  8(%ebp), %eax
for:    cmpl    12(%ebp), %edx
        jge             avanti
        movl    (%eax, %edx, 4), %ebx
        cmpl    (%eax, %ecx, 4), %ebx
        jge             l2
        movl  %edx, %ecx
l2:     incl  %edx
        jmp             for
avanti: movl    (%eax, %ecx, 4), %ebx
        movl  16(%ebp), %esi
        movl    (%eax, %esi, 4), %edi
        movl    %edi, (%eax, %ecx, 4)
        movl  %ebx, (%eax, %esi, 4)
        popl    %edi
        popl    %esi
        popl    %edx
        popl    %ecx
        popl    %ebx
        popl    %eax
        leave
        ret
```

2. (a)
```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>

int max;
int num;

void set_num(int signo)
{
        num = rand() % max + 1;
}
```

```
void stop(int signo)
{
        exit(0);
}

int main(int argc, char* argv[])
{
        if (argc != 2) {
                fprintf(stderr, "Uso: %s <max>\n", argv[0]);
                exit(1);
        }

        if (sscanf(argv[1], "%d", &max) != 1 || max <= 0) {
                fprintf(stderr, "l'argomento deve essere un numero maggiore di 0");
                exit(1);
        }

        srand(time(0));
        set_num(0);

        if (signal(SIGUSR1, set_num) == SIG_ERR) {
                perror(argv[0]);
                exit(1);
        }
        if (signal(SIGUSR2, stop) == SIG_ERR) {
                perror(argv[0]);
                exit(1);
        }

        for (;;) {
                printf("%d\n", num);
                fflush(stdout);
                sleep(1);
        }

        return 0;
}
```

(b)
```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <signal.h>
#include <errno.h>

#define BUFSIZE 100

int main(int argc, char *argv[])
{
        int pid, target, r;
        int p[2];

        if (argc != 3) {
```

```
        fprintf(stderr, "Uso: %s <target> <max>\n", argv[0]);
        exit(1);
}

if (sscanf(argv[1], "%d", &target) != 1 || target <= 0) {
        fprintf(stderr, "target deve essere maggiore o uguale a 0\n");
        exit(1);
}

if (pipe(p) < 0) {
        perror(argv[0]);
        exit(1);
}

switch (pid = fork()) {
case -1:
        perror(argv[0]);
        exit(1);
case 0:
        close(1);
        dup(p[1]);
        close(p[0]);
        close(p[1]);
        execl("tick", "tick", argv[2], NULL);
        perror("tick");
        exit(1);
default:
        close(0);
        dup(p[0]);
        close(p[0]);
        close(p[1]);
        for (;;) {
                if (scanf("%d\n", &r) < 0) {
                        if (errno != 0) perror(argv[0]);
                        exit(1);
                }
                printf("ricevuto %d\n", r);
                if (r >= target) {
                        printf("ok\n");
                        if (kill(pid, SIGUSR2) < 0) {
                                perror(argv[0]);
                                exit(1);
                        }
                        break;
                }
                printf("riprovo...\n");
                if (kill(pid, SIGUSR1) < 0) {
                        perror(argv[0]);
                        exit(1);
                }
        }
}
wait(NULL);
```

```
        return 0;
}
```