Soluzioni della Prova Scritta di Sistemi di Elaborazione Ingegneria delle Telecomunicazioni

Ing. G. Lettieri, Ing. A. Vecchio

2 luglio 2010

1. (a) Il programma legge parole dallo standard input fino a quando l'utente non inserisce una parola uguale a ".". Quindi stampa il numero di caratteri della parola più lunga ricevuta in ingresso e termina la propria esecuzione.

   (b) Una possibile traduzione è la seguente:

```
.data
formato: .asciz "%s"
fin_str: .asciz "."
.text
.global f2
f2:
  pushl %ebp
  movl  %esp, %ebp

  pushl %ebx
  pushl %esi

  subl  MAXL, %esp
  movl  %esp, %ebx
  movl  $0, %esi
ciclo:
  pushl %ebx
  pushl $formato
  call  scanf
  addl  $8, %esp
  pushl %ebx
  call  f1
  addl  $4, %esp
  cmpl  %esi, %eax
  jle   avanti
  movl  %eax, %esi
avanti:
  pushl $fin_str
  pushl %ebx
  call  strcmp
  addl  $8, %esp
  cmpl  $0, %eax
  jne   ciclo

  movl  %esi, %eax
  popl  %esi
  popl  %ebx
  leave
  ret
```

2. (a)
```
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <string.h>

int main(int argc, char* argv[])
{
        int i, tot = 0;

        if (argc < 2) {
                fprintf(stderr, "Uso: %s <dir>...\n", argv[0]);
                exit(1);
        }
```

```
        for (i = 1; i < argc; i++) {
                DIR *d;
                struct stat st;
                struct dirent *e;

                if ( stat(argv[i], &st) < 0) {
                        perror(argv[2]);
                        continue;
                }

                if (!S_ISDIR(st.st_mode)) {
                        fprintf(stderr, "%s non e' una directory\n", argv[i]);
                        continue;
                }

                if ( !(d = opendir(argv[i])) ) {
                        perror(argv[i]);
                        continue;
                }

                while ( e = readdir(d) ) {
                        if (e->d_name[0] == '.' &&
                            strcmp(e->d_name, ".") != 0 &&
                            strcmp(e->d_name, "..") != 0)
                                tot++;
                }
                closedir(d);
        }
        printf("%d\n", tot);
        return 0;
}
```

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

#define MAX_DIRS 100
#define MAX_LINE 1024

int main(int argc, char* argv[])
{
        char dirs[MAX_DIRS][MAX_LINE];
        char* c_argv[MAX_DIRS + 2];
        int c_argc, fd[2];

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

        if (pipe(fd) < 0) {
```

```
                perror(argv[0]);
                exit(1);
        }

        switch (fork()) {
        case -1:
                perror(argv[0]);
                exit(1);
        case 0:
                close(1);
                dup(fd[1]);
                close(fd[0]);
                close(fd[1]);
                execlp("find", "find", argv[1], "-type", "d", NULL);
                perror("find");
                exit(1);
        default:
                break;
        }

        switch (fork()) {
        case -1:
                perror(argv[0]);
                exit(1);
        case 0:
                close(0);
                dup(fd[0]);
                close(fd[0]);
                close(fd[1]);

                c_argv[0] = "contanascosti";
                c_argc = 1;
                while (c_argc - 1 < MAX_DIRS &&
                       (c_argv[c_argc] = fgets(dirs[c_argc - 1], MAX_LINE, stdin)) ) {
                        c_argv[c_argc][strlen(c_argv[c_argc]) - 1] = '\0';
                        c_argc++;
                }
                c_argv[c_argc] = NULL;
                execv("contanascosti", c_argv);
                perror("contanascosti");
                exit(1);
        }

        close(fd[0]);
        close(fd[1]);
        wait(NULL);
        wait(NULL);
        return 0;
}
```