Soluzioni della Prova Scritta di Sistemi di Elaborazione Ingegneria delle Telecomunicazioni

Ing. G. Lettieri, Ing. A. Vecchio

13 gennaio 2009

1. (a) Il programma legge tutte le righe del file il cui nome passato come argomento da riga di comando e, per ogni riga, stampa una parola a caso.

   (b) Una possibile traduzione è la seguente:

```
        .data
formato:        .asciz "%s\n"
        .set MAXP, 100
        .text
        .global f2
f2:     pushl %ebp
        movl  %esp, %ebp
        pushl %eax
        pushl %ebx
        pushl %ecx
        pushl %edx
        pushl %esi
        pushl %edi
        # spazio per buf
        subl  $MAXP, %esp
        # ebx come buf
        movl  %esp, %ebx
        # uso esi come i
        movl  $0, %esi
        # uso edi come x
        movl  $0, %edi
        movl  8(%ebp), %ecx
while:  cmpb  $'\n, (%ecx, %esi)
        je    avanti
        cmpb  $' ', (%ecx, %esi)
        jne   l1
        incl  %edi
l1:     incl  %esi
        jmp while
avanti: incl  %edi
        call  random
        # dentro eax c'e' il
        # valore prodoto da random
        movl  $0, %edx
        idiv  %edi
        pushl %edx
        pushl %ebx
        pushl 8(%ebp)
        call  f1
        addl  $12, %esp
        pushl %ebx
        pushl $formato
        call  printf
        addl  $8, %esp
        popl  %edi
        popl  %esi
        popl  %edx
        popl  %ecx
        popl  %ebx
        popl  %eax
        leave
        ret
```

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

#define MAXLINE 1024

int getmaxline(char *name)
{
```

```
        char line[MAXLINE];
        int max = 0, l;
        FILE *f;

        if ( !(f = fopen(name, "r")) ) {
                perror(name);
                return -1;
        }

        while ( fgets(line, MAXLINE, f) ) {
                l = strlen(line);
                if (l > max)
                        max = l;
        }
        fclose(f);
        return max;
}

int main(int argc, char* argv[])
{
        int max = 0, cur, l;
        char name[MAXLINE], name_m[MAXLINE];
        int tipo;
        struct stat st;
char *oo[] = { "name", "size", "line" };

        if (argc != 2) {
                fprintf(stderr, "Uso: %s name|size|line\n", argv[0]);
                exit(1);
        }

for (tipo = 0; tipo < 3; tipo++)
if (strcmp(oo[tipo], argv[1]) == 0)
break;
        if (tipo >= 3) {
                fprintf(stderr, "tipo sconosciuto: '%s'\n", argv[1]);
                exit(1);
        }

        while ( fgets(name, MAXLINE, stdin) ) {
                if ( (l = strlen(name)) < 1)
continue;
                name[l - 1] = '\0';
                if ( stat(name, &st) < 0 ) {
                        perror(name);
                        continue;
                }
                switch (tipo) {
                case 0:
                        cur = l;
                        break;
                case 1:
                        cur = st.st_size;
```

```
                        break;
                case 2:
                        if ( S_ISREG(st.st_mode) )
                                cur = getmaxline(name);
                        break;
                }

                if (cur > max) {
                        max = cur;
                        strcpy(name_m, name);
                }
        }
        printf("%s %d\n", name_m, max);
        return 0;
}
```

(b)
```
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[])
{
        int n = 0, i, ok = 0;
        int fd[2];
        struct stat st;

        for (i = 1; i < argc; i++) {
                if ( stat(argv[i], &st) < 0) {
                        perror(argv[i]);
                        continue;
                }
                if ( !S_ISDIR(st.st_mode) ) {
                        fprintf(stderr, "%s: non e' una directory\n", argv[i]);
                        continue;
                }

                if (pipe(fd) < 0) {
                        perror(argv[0]);
                        continue;
                }

                ok = 0;
                switch (fork()) {
                case -1:
                        perror(argv[0]);
                        break;;
                case 0:
                        close(1);
                        dup(fd[1]);
```

```
                        close(fd[0]);
                        close(fd[1]);
                        execlp("find", "find", argv[i], NULL);
                        perror("ls");
                        break;
                default:
                        ok = 1;
                        n++;
                        break;
                }

                if (ok) {
                        switch (fork()) {
                        case -1:
                                perror(argv[0]);
                                continue;
                        case 0:
                                close(0);
                                dup(fd[0]);
                                close(fd[0]);
                                close(fd[1]);
                                execl("biggest", "biggest", "size", NULL);
                                perror("biggest");
                                break;
                        default:
                                n++;
                                break;
                        }
                }
                close(fd[0]);
                close(fd[1]);
        }
        while (n) {
                wait(NULL);
                n--;
        }
        return 0;
}
```