

Seconda simulazione

```
1  #include<stdio.h>
2  #include<string.h>
3  #include<stdlib.h>
4  #include<unistd.h>
5  #include<stdbool.h>
6  #include<signal.h>
7  #include<termios.h>
8
9  #define ARGUMENTS 3
10 #define MAXCHILD 10
11
12 typedef enum {
13     KP_ECHO_OFF,
14     KP_ECHO_ON,
15 } kp_echo_t;
16
17 FILE * ptr=NULL;
18 int n_child=0,v_child[MAXCHILD],pid_server;
19
20 int keypress(const kp_echo_t echo) {
21     struct termios savedState, newState;
22     unsigned char echo_bit; // flag
23     int c;
24     if (-1 == tcgetattr(STDIN_FILENO, &savedState)) { return EOF; }; // error
25     newState = savedState;
26     if (KP_ECHO_OFF == echo) { echo_bit = ECHO; } else { echo_bit = 0; };
27     /* canonical input + set echo with minimal input as 1. */
28     newState.c_lflag &= ~(echo_bit | ICANON);
29     newState.c_cc[VMIN] = 1;
30     if (-1 == tcsetattr(STDIN_FILENO, TCSANOW, &newState)) { return EOF; }; // error
31     c = getchar(); /* block until key press */
32     if (-1 == tcsetattr(STDIN_FILENO, TCSANOW, &savedState)) { return EOF; }; // error
33     return c;
34 }
35
36 bool file_check(const char *path){
37     FILE *tmp=fopen(path,"r");
38     bool trovato=false;
39     if(ptr!=NULL){
40         trovato=true;
41     }
42     fclose(tmp);
43     return trovato;
44 }
45
46 void server_handler(int signo){
47     if(signo==SIGINT){
```

```

48     fprintf(ptr,"%d\n",n_child);
49     fflush(ptr);
50     exit(0);
51 }
52 if(signo==SIGUSR1){
53     if(n_child<MAXCHILD){
54         int child=fork();
55         if(child==0){
56             while(1){
57                 pause();
58             }
59         }
60         else if(child>0){
61             v_child[n_child]=child;
62             n_child++;
63             fprintf(ptr,"%d\n",child);
64             fflush(ptr);
65             printf("[server] %d\n",child);
66             fflush(stdout);
67         }
68         else{
69             exit(-99);
70         }
71     }
72 }
73 if(signo==SIGUSR2){
74     if(n_child>0){
75         n_child--;
76         int figlio=v_child[n_child];
77         fprintf(ptr,"-%d\n",figlio);
78         fflush(ptr);
79         printf("[server] %d\n",figlio);
80         fflush(stdout);
81     }
82     else{
83         fprintf(ptr,"-%d\n",0);
84         fflush(ptr);
85         printf("[server] %d\n",0);
86         fflush(stdout);
87     }
88 }
89 }
90
91 void write_pid(){
92     fprintf(ptr,"%d\n",getpid());
93     fflush(ptr);
94     printf("[server:%d]\n",getpid());
95     fflush(stdout);
96 }
97
98 int main(int argc, char ** argv) {
99     if(argc!=ARGUMENTS){
100         exit(-1);
101     }
102
103     if(strcmp(argv[1],"server")==0){

```

```

104     ptr=fopen(argv[2],"w");
105
106     if(ptr==NULL || !file_check(argv[2])){
107         exit(-2);
108     }
109
110     signal(SIGUSR1,server_handler);
111     signal(SIGUSR2,server_handler);
112     signal(SIGINT,server_handler);
113     write_pid();
114     while(1){
115         pause();
116     }
117 }
118 if(strcmp(argv[1],"client")==0){
119     while(ptr==NULL){
120         ptr=fopen(argv[2],"r");
121     }
122     fscanf(ptr,"%d\n",&pid_server);
123     char c;
124     int counter=0;
125     while (1) {
126         c = keypress(KP_ECHO_OFF);
127         if (c=='+') {
128             printf("PLUS\n");
129             if(counter<10){
130                 kill(pid_server,SIGUSR1);
131                 counter++;
132             }
133         }
134         if (c=='-') {
135             printf("MINUS\n");
136             if(counter>0){
137                 kill(pid_server,SIGUSR2);
138                 counter--;
139             }
140         }
141         if (c=='\n') {
142             printf("ENTER\n");
143             for(;counter>0;--counter){
144                 kill(pid_server,SIGUSR2);
145                 sleep(1);
146             }
147             kill(pid_server,SIGINT);
148             break;
149         }
150     }
151 }
152 return 0;
153 }

```