Seconda simulazione

```c
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<unistd.h>
#include<stdbool.h>
#include<signal.h>
#include<termios.h>

#define ARGUMENTS 3
#define MAXCHILD 10

typedef enum {
    KP_ECHO_OFF,
    KP_ECHO_ON,
} kp_echo_t;

FILE * ptr=NULL;
int n_child=0,v_child[MAXCHILD],pid_server;

int keypress(const kp_echo_t echo) {
    struct termios savedState, newState;
    unsigned char echo_bit; // flag
    int c;
    if (-1 == tcgetattr(STDIN_FILENO, &savedState)) { return EOF; }; // error
    newState = savedState;
    if (KP_ECHO_OFF == echo) { echo_bit = ECHO; } else { echo_bit = 0; };
    /* canonical input + set echo with minimal input as 1. */
    newState.c_lflag &= ~(echo_bit | ICANON);
    newState.c_cc[VMIN] = 1;
    if (-1 == tcsetattr(STDIN_FILENO, TCSANOW, &newState)) { return EOF; }; // error
    c = getchar(); /* block until key press */
    if (-1 == tcsetattr(STDIN_FILENO, TCSANOW, &savedState)) { return EOF; }; // error
    return c;
}

bool file_check(const char *path){
    FILE *tmp=fopen(path,"r");
    bool trovato=false;
    if(ptr!=NULL){
        trovato=true;
    }
    fclose(tmp);
    return trovato;
}

void server_handler(int signo){
    if(signo==SIGINT){
        fprintf(ptr,"%d\n",n_child);
        fflush(ptr);
        exit(0);
    }
    if(signo==SIGUSR1){
        if(n_child<MAXCHILD){
            int child=fork();
            if(child==0){
                while(1){
                    pause();
                }
            }
            else if(child>0){
                v_child[n_child]=child;
                n_child++;
                fprintf(ptr,"+%d\n",child);
                fflush(ptr);
                printf("[server] %d\n",child);
                fflush(stdout);
            }
            else{
                exit(-99);
            }
        }
    }
    if(signo==SIGUSR2){
        if(n_child>0){
            n_child--;
            int figlio=v_child[n_child];
            fprintf(ptr,"-%d\n",figlio);
            fflush(ptr);
            printf("[server] %d\n",figlio);
            fflush(stdout);
        }
        else{
            fprintf(ptr,"-%d\n",0);
            fflush(ptr);
            printf("[server] %d\n",0);
            fflush(stdout);
        }
    }
}

void write_pid(){
    fprintf(ptr,"%d\n",getpid());
    fflush(ptr);
    printf("[server:%d]\n",getpid());
    fflush(stdout);
}

int main(int argc, char ** argv) {
    if(argc!=ARGUMENTS){
        exit(-1);
    }

    if(strcmp(argv[1],"server")==0){
        ptr=fopen(argv[2],"w");

        if(ptr==NULL || !file_check(argv[2])){
            exit(-2);
        }

        signal(SIGUSR1,server_handler);
        signal(SIGUSR2,server_handler);
        signal(SIGINT,server_handler);
        write_pid();
        while(1){
            pause();
        }
    }
    if(strcmp(argv[1],"client")==0){
        while(ptr==NULL){
            ptr=fopen(argv[2],"r");
        }
        fscanf(ptr,"%d\n",&pid_server);
        char c;
        int counter=0;
        while (1) {
            c = keypress(KP_ECHO_OFF);
            if (c=='+') {
                printf("PLUS\n");
                if(counter<10){
                    kill(pid_server,SIGUSR1);
                    counter++;
                }
            }
            if (c=='-') {
                printf("MINUS\n");
                if(counter>0){
                    kill(pid_server,SIGUSR2);
                    counter--;
                }
            }
            if (c=='\n') {
                printf("ENTER\n");
                for(;counter>0;--counter){
                    kill(pid_server,SIGUSR2);
                    sleep(1);
                }
                kill(pid_server,SIGINT);
                break;
            }
        }
    }
    return 0;
}
```