```c
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/wait.h>
int main(int argc, char *argv[]){


        int pid;

        if(argc == 1){
                printf("sintassi errata\n");
                exit(1);
        }


        for( int i=1; i<argc; i++){
                pid=fork();
                if(pid==0){
                        printf("Processo creato: %d\n", getpid());
                        sleep(1);
                        printf("Messaggio: %s\n", argv[i]);
                        exit(0);
                }

                else{
                        wait(NULL);

                }

}
                exit(0);

}
```

```c
main(int argc, char *argv[]){

    int N;
    int pid=0;
    int status;

    if(argc == 1){
            printf("sintassi errata\n");
            exit(1);
    }

    if(N>10)
            exit(1);


    for( int i=1; i<argc; i++){
            pid=fork();

            if(pid==0){

                    int targa= atoi(argv[i]);
                    if(targa >= 1 && targa <=90){

                            printf("Codice valido associato al processo %d\n %s\n", getpid(), argv[i]);
                            exit(0);


                    }

                    else{

                            exit(0);
                    }
            }
            else{
                            pid=wait(&status);
                            printf("%d Creato figlio: %d\n", pid,getpid());
                            printf("Termina processo\n", WEXITSTATUS(status));
                    }
```

segnali.c

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

void sighandler(int sig) {
        char c;

        printf("\n");
        printf("Forse hai premuto per sbaglio Ctrl-C?\nVuoi realmente sospendere l'esecuzione [y/n]?");
        c=getchar();

        if(c=='y' || c=='Y')
                exit(0);

}


int main(int argc, char* argv[]) {
        int N = atoi(argv[1]);
        int count =0;

        while((N-1)>count) {
                if(signal(SIGINT, sighandler)) {
                        count++;
                }
                pause();

        }
        printf("\nRaggiunto N\n");

}
```

```c
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
#include <stdlib.h>
#define max 10
        void gestore_att(int sig){
                if (sig == SIGUSR1)
                        printf("Processo attivato: %d\n", getpid());
                else
                        printf("Processo non attivato: %d\n", getpid());
                        exit(0);
        }

        int main(int argc , char *argv[]){
        int pid[max], pf;
        int i, status;
        char request;

        int N = atoi(argv[1]);
        if (argc==1){
                printf("sintassi sbagliata!\n");
                exit(1);
        }
        for(i=0; i< N; i++){
                pid[i]=fork();
        if (pid[i]==0){
                signal(SIGUSR1, gestore_att);
                signal(SIGUSR2, gestore_att);
                pause();
    }
   else {
                printf("Creato processo figlio: %d\n", pid[i]);
                printf("Attivare il processo %d? [y/n]: ",pid[i]);
                request=getchar();
        if ((request=='Y') || (request=='y'))
           kill(pid[i], SIGUSR1);
           else
                kill(pid[i], SIGUSR2);
```

```
                signal(SIGUSR1, gestore_att);
                signal(SIGUSR2, gestore_att);
                pause();
    }
   else {
                printf("Creato processo figlio: %d\n", pid[i]);
                printf("Attivare il processo %d? [y/n]: ",pid[i]
                request=getchar();
        if ((request=='Y') || (request=='y'))
           kill(pid[i], SIGUSR1);
           else
                   kill(pid[i], SIGUSR2);

        request=getchar(); //serve a catturare l'andare a capo
    ///nel momento in cui si preme INVIO
    //subito dopo aver digitato il carattere 'y' o 'n'
        pf= wait(&status);
        }
 }
 exit(0);
} /* fine padre */
```