~/Desktop/a.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <wait.h>
#include <signal.h>
#include <sys/types.h>
#include "fibonacci.h" // int fib(int n)
#include "fattoriale.h" // int fatt(int n)

void decimo_fib_handler(int signo) {
    printf("il decimo numero di Fibonacci e': %d\n", fib(10));
}

void ctrl_c_handler(int signo) {
    signal(SIGINT, SIG_IGN);
    char c;
    printf("Continuare? (y/n) >");
    scanf("%c", &c);
    if (c == 'n') {
        raise(SIGKILL);
        // or exit(0)
    }
    // con qualcosa di diveso da n si continua
}

int main() {
    signal(SIGINT, SIG_IGN);
    pid_t F1, F2;
    F1 = fork();
    if (F1 < 0) { // controllo riuscita fork
        fprintf(stderr, "Can't fork\n");
        exit(EXIT_FAILURE);
    }
    if (F1 == 0) { // FIGLIO1
        for (int i = 0; i < 30; i++) {
            signal(SIGINT, decimo_fib_handler);
            printf("[F1]: fib(%d)=%d\n", i, fib(i));
            sleep(2);
        }
        exit(EXIT_SUCCESS);
    }
    else { // PADRE
        F2 = fork();
        //controllo riuscita della fork
        if (F2 == 0) {
            for (int i = 0; i < 20; i++) {
                signal(SIGINT, ctrl_c_handler);
                printf("[F2]: fatt(%d)=%d\n", i, fatt(i));
                sleep(2);

            }
            exit(EXIT_SUCCESS);
        }
```

```c
        else { // PADRE
            waitpid(F1, NULL, 0);
            waitpid(F2, NULL, 0);
            printf("[P]: i due processi sono terminati\n");
            printf("[P]: la somma dei loro PID e': %d\n", (F1+F2));

        }
    }
    return 0;
}

```