

```

#include <stdio.h>
#include <sys/errno.h>
#include <sys/wait.h>
#include <signal.h>
#include <string.h>
#define MAX 50

//funzionu di gestione
void figlio (int i, char car, char *nomefile[]);
void handler (int signo);

//parametri di esecuzione
char buff[DIM];
...

//altre variabili globali
int pid[MAX];
int fd, pipes[MAX][2];
...

main (int argc, char *argv[]) {
int i, status, pid, ppid;
...

//controllo dei parametri
if (argc!=3) {
    perror ("Errore nella sitassi");
    exit(-2);
}

//inizializzazione delle variabili
strcpy (nomefile, argv[1]); = read(0,buff,DIM);
N = atoi(argv[2]);
car = argv[3][0];

ppid = getpid();
...

//apertura pipe
for (i=0;i<numfigli;i++) {
    if (pipe(pipes[i])<0) {
        perror ("Errore nella creazione della pipe");
        exit(-3);
    }
    else {
        printf ("pipes[%d][0] = %d\t", i,pipes[i][0]);
        printf ("pipes[%d][1] = %d\n", i,pipes[i][1]);
    }
}

//apertura del file di output (fd ereditato dai figli)
if ((fd_out=open(argv[argc-1],O_RDWR | O_CREAT, 0777))<0) {
    perror ("Errore nell'apertura del file di output");
    exit(-4);
}
else printf ("File %s aperto correttamente\n", argv[argc-1]);

//ciclo di generazione dei figli
for (i=0;i<numfigli;i++) {
    pid=fork();

```

```

    if (pid<0) {
        perror ("Errore nella generazione del figlio");
        exit(-5);
    }
    else if (pid==0) {
        figlio (i,argv[i][0], *nomefile);
        exit(0);
    }
    else printf ("P0 (%d): creazione del figlio P%d (%d)\n",ppid,i+1,pid);
}

//aggancio dei segnali handler del padre
signal (SIGUSR1, handler);
signal (SIGUSR2, handler);

//chiusura dei descrittori non utilizzati dal padre
for (i=0;i<numfigli;i++) {
    close (pipes[i][0]);
    close (pipes[i][1]);
}

//ciclo di attesa della terminazione dei figli
for (i=0;i<numfigli;i++) {
    pid=wait(&status);
    if (char(status)!=0)
        printf ("Terminazione involontaria di %d per segnale %d\n",pid,(char)
status);
    else
        printf ("Terminazione volontaria di %d con stato %d\n",pid,(char)
status>>8);
}

exit(0);
}

void figlio (int i, char car, char *nomefile[]) {
}

void handler (int signo) {
}

//alla fine di una pipe, quando ho chiuso tutti i fd, exit(0)
//con le kill(pid,SIGUSR1), pid e ppid devono sempre essere variabili globali

```