#include<unistd.h>

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<fcntl.h>

#include<errno.h>

#include<string.h>

#include <dirent.h>

#include<signal.h>

#include<signal.h>

/\*

gcc -g chequeaArchivo.c - o CHE1

mac-->> ./CHE1 ~/Desktop/prueba/archivo1.txt ~/Desktop/prueba/archivo2.txt

ubuntu-->> ./CHE2 ./home/miguel/Escritorio/prueba/archivo1.txt ./home/miguel/Escritorio/prueba/archivo2.txt

iMPRIMIRA EL PADRE CUANDO ALGUNO DE LOS ARCHIVOS HAYAN SIDO MODIFICADOS

void manejador(int sig)

{

wait(estado);//para no quedar hijos zombie, null si no te interesa como acaba el hujo

printf("&d",var\_global);

}

struct sigaction sa;

sa.sa\_handler=manejador;

if(sigaction(SIGCHLD,&sa,NULL)<0) perror("error signation"),exit(1);

\*/

int main(int argc, char const \*argv[])

{

if(argc!=3)

{

perror("Error numero argumentos");

exit(1);

}

if(setvbuf(stdout,NULL,\_IONBF,0))

{

perror("ERROR EN EL SETBUFF");

}

int t;

pid\_t PID;

struct stat atributos;

/\*

/////////////////////////me lo puedo ahorrar//////////////////////////////////////

if(stat(argv[1],&atributos)<0)

{

perror("Error AL acceder al los atributos ARGV[1]");

exit(1);

}

if(!(S\_ISREG(atributos.st\_mode)))

{

perror("NO ES UN ARCHIVO REGULAR\n");

exit (-1);

}

/////////////////////////////////////////////////////////////

if(stat(argv[2],&atributos)<0)

{

perror("Error AL acceder al los atributos ARGV[2]");

exit(1);

}

if(!(S\_ISREG(atributos.st\_mode)))

{

perror("NO ES UN ARCHIVO REGULAR\n");

exit (-1);

}

\*/

//////////////////////////////////////////////////////////////

if( mkfifo("datos",0777)<0)//si no es puntero debo poner &

{

perror("Error AL CREAR EL CAUCE");

exit(1);

}

/\*if(pipe(datos)<0)

{

perror("error al crear el pipe");

exit(1);

}

\*/

///////////////////////////// HIJO1 //////////////////////////////////////////////////

if((PID=fork())<0)

{

perror("error al crear HIJO1");

exit(1);

}

if (PID==0)

{

//abro para lectura y escritura

if((t=open("datos",O\_RDWR))<0) //ANTE ERROR MEJOR O\_RDWR

{

perror("error al abrir el cauce para ESCRIBIR1");

exit(1);

}

if((dup2(t, STDOUT\_FILENO))<0)

{

perror("error DUP2...1");

exit(1);

}

execlp("./CHE1","./CHE1",argv[1],NULL);

perror("Error en el EXECLP");

exit(1);

}

////////////////////////////HIJO2//////////////////////////////////////////////////

else

{

if((PID=fork())<0)

{

perror("error al crear HIJO1");

exit(1);

}

if (PID==0)

{

if((t=open("datos",O\_RDWR))<0) //ANTE ERROR MEJOR O\_RDWR

{

perror("error al abrir el cauce para ESCRIBIR HIJO2");

exit(1);

}

if((dup2(t, STDOUT\_FILENO))<0)

{

perror("error DUP2...2");

exit(1);

}

execlp("./CHE1","./CHE1",argv[2],NULL);

perror("Error en el EXECLP......2");

exit(1);

}

////////////////////////////PARES//////////////////////////////////////////////////

else

{

printf("weee\n");

if((t=open("datos",O\_RDWR))<0)

{

perror("error al abrir el cauce para LEER padre");

exit(1);

}

char cadena[11];

int entero;

while((entero=read(t,cadena,sizeof(char)\*11))>0)

{

//atoi(cadena);

//global=atoi(cadena);

printf("READ!\n");

if(write(STDOUT\_FILENO,cadena,sizeof(char)\*entero)<0)

{

perror("Error a la hora de ESCRIBIR");

exit(1);

}

}

//kill(SIGINT,hijo1);

}

}

return 0;

}