//EJEMPLO 1

#include <stdio.h>

#include <stdlib.h>

void main (void){

printf ("Ejemplo de uso de system ():");

printf ("\n\tListado del directorio actual y envió a un fichero:");

system ("ls > ficsalida" );

printf ("\n\tAbrimos con el gedit el fichero. . . ");

printf ("%d",system ("gedit ficsalida"));

printf ("\n\tEste comando es erróneo: %d",system("ged"));

printf ("\nFin de programa ....\n");

}

//EJEMPLO 2

#include <unistd.h>

#include <stdio.h>

void main(void){

printf("Ejemplo de uso de exec():");

printf("Los archivos en el directorio son:\n");

execl("/bin/ls", "ls", "-1", (char \*)NULL);

printf("¡¡¡ Esto no se ejecuta !!!\n");

}

//EJEMPLO 3

#include <unistd.h>

#include <stdio.h>

void main(void){

pid\_t id\_pactual,id\_padre;

id\_pactual=getpid();

id\_padre=getppid();

printf("PID de este proceso: %d\n",id\_pactual);

printf("PID del proceso padre: %d\n",id\_padre);

}

//EJEMPLO 4

#include <stdlib.h>//Para el exit del fork

#include <unistd.h>

#include <stdio.h>

#include <sys/wait.h>

void main(void){

pid\_t pid, hijo\_pid;

pid=fork();

if (pid==-1){ //Ha ocurrido un error

printf("No se ha podido crear el proceso hijo...");

exit(-1);

}

else if (pid == 0 ){ //Nos encontramos en Proceso hijo

printf("Soy el proceso hijo \n\t Mi PID es %d, El PID de mi padre es: \

%d.\n pulse intro para continuar …", getpid(), getppid() );

getchar();

}

else {//Nos encontramos en Proceso padre

hijo\_pid = wait(NULL); //espera la finalización del proceso hijo

printf("Soy el proceso padre:\n\t\

Mi PID es %d, El PID de mi padre es: %d.\n\t \

Mi hijo: %d terminó.\n", getpid(), getppid(), pid);

}

}

//EJEMPLO 5

#include <stdlib.h>

#include <unistd.h>

#include <stdio.h>

#include <sys/wait.h>

void main (void){

pid\_t pid, hijo\_pid, pid2, hijo2\_pid;

pid = fork(); //Soy el Abuelo, creo a Hijo

if (pid == -1 ){ //Ha ocurrido un error

printf ("No se ha podido crear el proceso hijo...");

exit(-1);

}

else if (pid == 0 ){ //Nos encontramos en Proceso hijo

pid2 = fork(); //Soy el Hijo, creo a Nieto

switch(pid2){

case -1: // error

printf ("No se ha podido crear el proceso nieto. . .");

exit(-1);

break;

case 0: // proceso NIETO

printf("\t\tSoy el proceso nieto %d; Mi padre es = %d \n", getpid(), getppid());

break;

default: // proceso padre

hijo2\_pid=wait(NULL);

printf("\tSoy el proceso %d, hijo del %d.\n", getpid(), getppid());

printf("\tMi hijo: %d terminó.\n", hijo2\_pid);

}

}

else{ //Nos encontramos en Proceso abuelo

hijo\_pid = wait(NULL); //espera la finalización del proceso hijo

printf("Soy el proceso abuelo: %d, Mi hijo: %d terminó.\n", getpid(), pid);

}

}

//EJEMPLO 6

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/wait.h>

void main(void) {

pid\_t pid;

pid = fork();

if (pid == -1) {

printf ("No se ha podido crear el proceso hijo...");

exit(-1);

}

else if (pid == 0) {

while (1) {

printf("Soy el hijo %d, mi padre es %d\n",getpid(), getppid());

sleep(1);

}

}

else {

printf("Soy el padre, mi PID es %d\n", getpid());

sleep(5);

printf("El proceso padre muere\n");

}

}

//EJEMPLO 7 , VERSIÓN 1

#include <unistd.h>

#include <stdio.h>

void main(void){

pid\_t pid;

pid = fork();

if (pid == -1) {

printf ("No se ha podido crear el proceso hijo...");

}

else{

while(1){

printf("Proceso con PID = %d se está ejecutando \n", getpid());

sleep(1);

}

}

}

//EJEMPLO 7 , VERSIÓN 2

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

#include<stdbool.h>

#include <sys/wait.h>

void main(void){

pid\_t pid;

pid = fork();

if (pid == -1) {

printf ("No se ha podido crear el proceso hijo...");

exit (-1);

}

else if (pid > 0) {

bool bandera = true;

while(1){

printf("Proceso padre con PID = %d se está ejecutando \n", getpid());

sleep(1);

if(bandera){

wait(NULL);

bandera = false;

}

}

}

else{

printf("Proceso hijo con PID = %d se está ejecutando \n", getpid());

}

}

//EJEMPLO 8 -> Cerrando ficheros

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

void main (){

int fd[2];

char buffer[30];

pid\_t pid;

pipe (fd); //se crea el pipe

pid = fork (); //se crea el proceso hijo

switch (pid){

case -1: //ERROR

printf ("NO SE HA PODIDO CREAR HIJO...");

exit (-1);

break;

case 0:

close(fd[0]);

printf ("El HIJO escribe en el pipe...\n");

write (fd[1], "Hola papi", strlen("Hola papi"));

close(fd[1]);

break;

default: //PADRE

close(fd[1]);

sleep(1);

printf ("El PADRE lee del pipe...\n");

read (fd[0], buffer, 10);

close(fd[0]);

printf ("\tMensaje leido: %s\n", buffer);

break;

}

}

//EJEMPLO 9

#include <stdlib.h>

#include <unistd.h>

#include <stdio.h>

#include <string.h>

#include <sys/types.h>

#include <sys/wait.h>

//PRIMERO-SEGUNDO-TERCERO

void main () {

pid\_t pid, segundo\_pid, pid2, tercero\_pid;

int fd1[2];

int fd2[2];

char saludoPrimero[] = "Saludos del primer proceso.\0";

char saludoSegundo[] = "Saludos del segundo proceso \0";

char saludoTercero[] = "Saludos del tercer proceso\0";

char buffer[80] = "";

pipe (fd1);

pipe (fd2);

pid = fork (); //Soy el primer proceso, creo al segundo

if (pid == -1){ //Ha ocurrido un error

printf ("No se ha podido crear el segundo proceso...");

exit (-1);

}

if (pid == 0){ //NOS ENCONTRAMOS EN EL SEGUNDO PROCESO

pid2 = fork (); //Soy el segundo proceso, creo al tercero

switch (pid2){

case -1: // error

printf ("No se ha podido crear el tercer proceso ...");

exit (-1);

break;

case 0: //NOS ENCONTRAMOS EN EL TERCER PROCESO

close (fd2[1]);

read (fd2[0], buffer, sizeof (buffer)); //leo el pipe

close (fd1[0]);

printf ("\t\tEl tercer proceso RECIBE MENSAJE del segundo proceso: %s\n", buffer);

printf ("\t\tEl tercer proceso ENVÍA MENSAJE al segundo proceso.\n");

write (fd1[1], saludoTercero, strlen (saludoTercero));

close (fd1[1]);

break;

default: //NOS ENCONTRAMOS EN EL SEGUNDO PROCESO

close (fd1[1]);

read (fd1[0], buffer, sizeof (buffer));

close (fd2[0]);

printf ("\tEl segundo proceso RECIBE MENSAJE del primer proceso: %s\n", buffer);

printf ("\tEl segundo proceso ENVÍA MENSJE al tercer proceso\n");

write (fd2[1], saludoSegundo, strlen (saludoSegundo));

tercero\_pid = wait (NULL); //espero al hijo

read (fd1[0], buffer, sizeof (buffer));

close (fd1[0]);

printf ("\tEl segundo proceso RECIBE MENSAJE del tercer proceso: %s\n", buffer);

printf ("\tEl segundo proceso ENVÍA MENSAJE al primer proceso\n");

write (fd2[1], saludoSegundo, strlen (saludoSegundo));

close (fd2[1]);

}

}

else{ //NOS ENCONTRAMOS EN EL PRIMER PROCESO

close (fd1[0]);

printf ("El primer proceso ENVÍA MENSAJE al segundo proceso\n");

write (fd1[1], saludoPrimero, strlen (saludoPrimero));

close (fd1[1]);

segundo\_pid = wait (NULL); //espera la finalizacion del hijo

close (fd2[1]);

read (fd2[0], buffer, sizeof (buffer));

close (fd2[0]);

printf ("El primer proceso RECIBE MENSAJE del segundo proceso: %s\n", buffer);

}

exit (0);

}

//Ejemplo 10\_E

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <fcntl.h>

void main (){

int fp;

char saludo[] = "Un saludo!!!\n";

fp = open ("FIF02", 1);

if (fp == -1){

printf ("ERROR AL ABRIR EL FICHERO...");

exit (-1);

}

printf ("Mandando información al FIFO...\n");

write (fp, saludo, strlen (saludo));

close (fp);

}

//Ejemplo 10\_L

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/stat.h>

#include <fcntl.h>

void main (void){

int fp;

int bytesLeidos;

char buffer[5] = "";

if (mknod ("FIF02", S\_IFIFO | 0666, 0) == -1){

printf ("HA OCURRIDO UN ERROR ...\n");

exit (-1);

}

while (1){

fp = open("FIF02", 0);

bytesLeidos = read (fp, buffer, sizeof(buffer));

printf ("OBTENIENDO Información...\n");

//La información se lee así para evitar que quede algo en el FIFO,

//si estamos seguros del tamaño estas líneas código son inecesarias.

while (bytesLeidos != 0)

{

printf ("%s", buffer);

bytesLeidos = read (fp, buffer, sizeof(buffer));

}

close (fp);

}

}

//Ejemplo 11

#include <stdio.h>

#include <signal.h>

#include <stdlib.h>

#include <fcntl.h>

#include <unistd.h>

/\* gestión de señales en proceso HIJO \*/

void manejador (int segnal) {

printf ("Hijo recibe señal..%d\n", segnal);

}

void main (){

int pid\_hijo;

pid\_hijo = fork (); //creamos hijo

switch (pid\_hijo){

case -1:

printf ("Error al crear el proceso hijo...\n");

exit (-1);

case 0: //HIJO

signal (SIGUSR1, manejador); //MANEJADOR DE SEÑAL EN HIJO

while (1){

system ("echo trabajando... >> ficsalida.txt");

sleep(1);

};

break;

default: //PADRE envia 2 señales

sleep (1);

kill (pid\_hijo, SIGUSR1); //ENVÍA SEÑAL AL HIJO

sleep (1);

kill (pid\_hijo, SIGUSR1); //ENVÍA SEÑAL AL HIJO

sleep (10);

kill(pid\_hijo, SIGTERM);

break;

}

}