#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <unistd.h>

#include <signal.h>

/\* Jaime Orlando Lòpez Ramos - A01374696

LAB 7 Sistemas Operativos

Objetivo: Aprender acerca del manejo de señales.

\*/

// Handler para SIGINT en el proceso padre

void catchInt(int sig\_num){

printf("Parent Process Received SIGINT \n");

fflush(stdout);

}

// Handler para SIGINT en el proceso hijo

void catchIntChild(){

printf("Child process received SIGINT \n");

fflush(stdout);

}

// Handler para SIGUSR1 (Manejada por el proceso hijo)

void catchUsr1(int sig\_num){

printf("Child process receive SIGUSR1\n");

}

// Handler para SIGUSR2 (Manejada por el proceso hijo)

void catchUsr2(int sig\_num){

printf("Child process received SIGUSR2\n");

}

// Funciòn main

int main(int argc, char\*\* argv){

bool seguirCorriendo = true; // Variable para determinar si el programa debe continuar su ejecuciòn

int input\_usuario; // Variable para guardar la decisiòn del usuario

int pid = fork(); // Llamamos a fork() para crear un proceso hijo

if(pid>0){

signal(SIGINT, catchInt); // El proceso padre se encarga de manejar las señales de SIGINT

// Entramos a un ciclo donde el usuario puede mandar a llamar algunas señales

while(seguirCorriendo){

printf("Pick signal to send: 1(SIGINT) 2(SIGUSR1) 3(SIGUSR2) 4(Exit):\n");

fflush(stdin);

scanf("%d", &input\_usuario);

// Dependiendo la selecciòn del usuario es la señal que se manda a llamar.

switch(input\_usuario){

case 1:

printf("Parent process sending SIGINT\n");

kill(pid, SIGINT);

break;

case 2:

printf("Parent process sending SIGUSR1\n");

kill(pid, SIGUSR1);

break;

case 3:

printf("Parent process sending SIGUSR2\n");

kill(pid, SIGUSR2);

break;

case 4:

kill(pid, SIGTERM);

seguirCorriendo = false;

break;

default:

printf("Input not recognized\n");

} // Switch Case ends here

sleep(0.6); // El proceso padre duerme 0.6 segundos para que el proceso hijo tenga suficiente tiempo de responder a la señal que se mandò

}

printf("Thanks for using the program. Terminating now, XOXO\n");

}

else{

// Asignamos a cada señal que será manejada por el proceso hijo la función que la manejará

signal(SIGINT, catchIntChild);

signal(SIGUSR1, catchUsr1);

signal(SIGUSR2, catchUsr2);

while(1); // Ciclo infinito para que el proceso hijo esté atento a cualquier señal

}