**Le binome:**

**Nom:Belharrat**

**Prenom:yanis zidane**

**Matricule:191931085131**

**Nom:Doudou**

**Prenom:amir**

**Matricule:191931081313**

**Projet de Systeme d'exploitation:**

Les captures d'ecran d'execution sont a la fin

bonne lecture.

Dans ce programme, on a un programme de fond qui marche dans une boucle infini,et 2 processus fils qui affichent a tour de role leur pid plus qlqs informations.

d'abord on a commence par utilise la primitive fork() pour cree 2 processus fils F1 et F2,tout ce programme de fond fonctionne dans une boucle infini while(1)

on a declare les bibliotheques dans on aura besoin pour l'utilisation des signaux et les affichages

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <stdlib.h>

#include <stdio.h>

#include <signal.h>

apres la creation des fils F1 et F2 avec fork();

on appelle avec le signal SIGUSR1 la procedure Hand\_F1

void Hand\_F1 (int sig) {

proc1id=getpid();

printf("processus1\_pid=%d : mesure 1\n",proc1id);

signal(SIGALRM,killPrecessus);

alarm(60);

//1er arret 1min=60secondes;

}

qui nous permet d'afficher le 1er affichage demande.

aussi pour Hand\_F2

void Hand\_F2 (int sig) {

proc2id=getpid();

printf("processus2\_pid=%d : mesure\n",proc2id);

signal(SIGALRM,killPrecessus);

alarm(60);//3eme arrer 1min=60secondes

}

pour l'attente entre les affichages, on a utilise la primitive alarm() qui utilise un timer,qui a la fin envoie un signal au programme,nous avons utilise une fonction handeler pour catch ce signal et Signal SIGCONT vers la procedure HAND P

void Hand\_P (int sig) {

compteur++;

if (compteur % 2 == 1) kill(pid2, SIGUSR2);

else kill(pid1, SIGUSR1);

}

dans la procedure Hand\_P on utilise le resultat du modulo pour alterner entre le fils 1 et le fils 2 et permettre de run a l'infini(ou jusqu'a l'arret du pc).

On a aussi catch le signal SIGINT pour arreter le programme avec 3 cliques de control+C avec une fonction handeler,

void INThandler(int signum)

{

if (signum == SIGINT) {

if (infini3){

if(infini2){

infini1 =1;}

infini2 = 1;}

infini3 = 1;

}

}

celle ci rends touts les whiles dans int main a 0 qui termine touts les processus actifs.

on a aussi utilise ces 2 fonctions handlers pour catchs les signaux SIGALRM

void waitHandler(int signum){

kill(getppid(), SIGCONT);

}

void killPrecessus(int signum){

i=0;

i++;

signal(SIGALRM,waitHandler);

if(i%2==1){if(proc1id!=0){printf("processus1\_pid=%d : mesure 2\n",proc1id);}

alarm(120);}//2eme arret 2min=120 secondes

else alarm(0);}

qui nous facilite la tache en cas de non disponibilite de sleep() ou pause()