# 1516 L2 SEPS TDTP 4– Signaux Unix

## I] Connexion



Nous pouvons voir sur l’image précédente , la connexion établie de la session et la création des différents dossiers demandés.

## II] Les signaux

### 1) Signaux.

Les signaux rapportent au programme l’occurrence d’un evenement exceptionnel. Parmi les raisons

qui peuvent entrainer la *génération* ou encore la *levée* d’un signal,

### 2) Envoi.

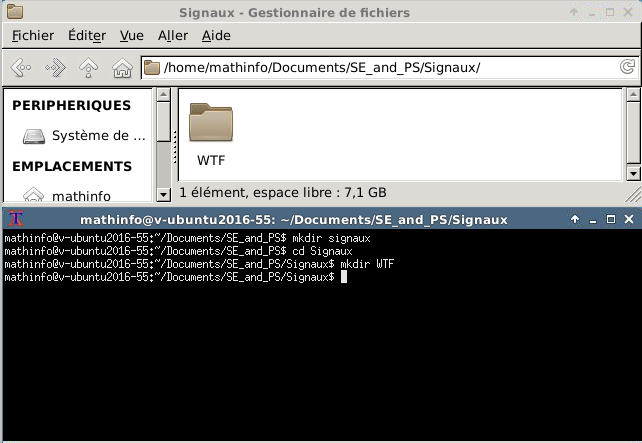
Les signaux peuvent etre envoyes dans un processus soit a partir de la fonction *raise(int SIG\_TYPE)*,pour envoyer le signal au processus appelant, ou grace a la fonction *kill(int pid, int SIG\_TYPE)*, qui permet de specifier le PID du processus destinataire.

### 3) Interception

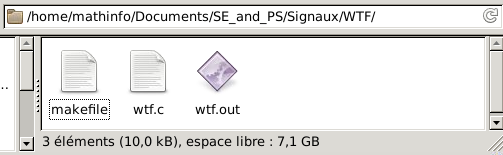
Le mode d’interception le plus direct se fait avec la fonction *signal(int SIG\_TYPE, void function)*, quipermet de declarer la fonction *function* comme *callback* pour le signal specifie. Si le signal specifie est SIG\_IGN, alors aucun callback ne sera execute a l’interception.

III] WTF ?

### Dans le repertoire "/home/mathinfo2014/Documents/SE\_and\_PS/Signaux", creez un repertoire "WTF".



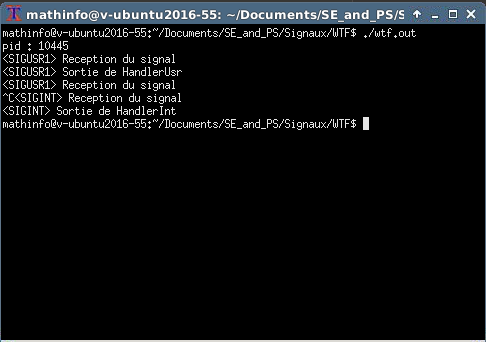
### Création du fichier du nom de wtf et du makefile



### Contenu du makefile

Z:\Documents\progra_system\1516_L2_SEPS_TDTP_4_STEGEN\Picture\e2.4.PNG

### Démonstartion du programme



### Code du programme

#include<stdio.h>

#include<stdlib.h>

#include<signal.h>

#include<string.h>

voidHandlerInt (int);

voidHandlerUsr (int);

int main (){

structsigactionSigAction;// Variable contenant les info pour le signal

printf("pid : %ld\n",getpid());

SigAction.sa\_handler = HandlerInt;// création du 1er signal (SIGINT)

sigemptyset(&SigAction.sa\_mask);

SigAction.sa\_flags = 0;

if(sigaction(SIGINT,&SigAction,NULL) == -1)

perror("Erreur d'armement du signal SIGINT");

SigAction.sa\_handler = HandlerUsr;// création du 2ème signal (SIGUSR1)

sigemptyset(&SigAction.sa\_mask);

SigAction.sa\_flags = 0;

if(sigaction(SIGUSR1,&SigAction,NULL) == -1)

perror("Erreur d'armement du signal SIGUSR1");

while(1) kill(getpid(),SIGUSR1);// Appel du signal SIGUSR1

exit(0);

}

voidHandlerInt (intsig){

printf("<SIGINT>Reception du signal\n");

printf("<SIGINT> Sortie de HandlerInt\n");

exit(0);

}

voidHandlerUsr (intsig){

printf("<SIGUSR1>Reception du signal\n");

sleep(5);

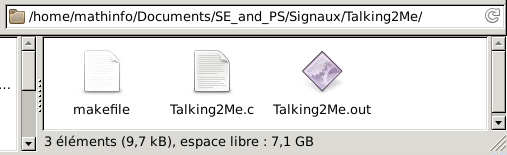
printf("<SIGUSR1> Sortie de HandlerUsr\n");

}

## IV] Are youtalking to me ?

### Dans le repertoire "/home/mathinfo2014/Documents/SE\_and\_PS/Signaux", creez un repertoire "Talking2Me".Z:\Documents\progra_system\1516_L2_SEPS_TDTP_4_STEGEN\Picture\e4.1.PNG

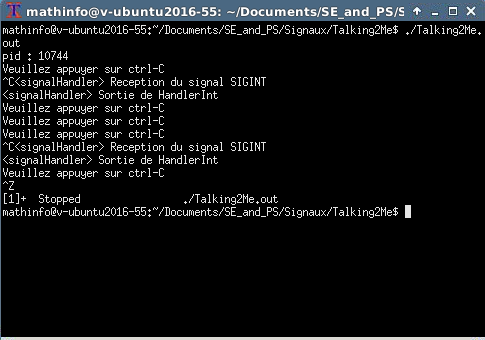
### Creez un fichier " talking2Me.c" et un fichier "makefile".



### Corrigez le makefile pour compiler Talking2Me.c.

Z:\Documents\progra_system\1516_L2_SEPS_TDTP_4_STEGEN\Picture\e4.3.PNG

### Dans le fichier "Talking2Me.c", ecrivez un programme en C qui intercepte le signal SIGINT (genere par Ctrl-c), et affiche un message toutes les secondes pendant quinze secondes invitant a effectuer la combinaison Ctrl-c au clavier. Le callback de SIGINT sera intitule ≪voidsignalHandler(intsignum) ≫ et affichera un message pour acquitter son appel.



### Code du programme

//Includes habituels voir exo précédent

voidsignalHandler (int);

int main (){

int i;

structsigactionSigAction;

printf("pid : %ld\n",getpid());

SigAction.sa\_handler = signalHandler;

sigemptyset(&SigAction.sa\_mask);

SigAction.sa\_flags = 0;

if(sigaction(SIGINT,&SigAction,NULL) == -1)

perror("Erreur d'armement du signal SIGINT");

for(i=0;i<15;i++){

sleep(1);

printf("Veuillez appuyer sur ctrl-C\n");

}

exit(0);

}

voidsignalHandler (intsignum){

switch(signum){

case SIGINT:

printf("<signalHandler>Reception du signal SIGINT\n");

printf("<signalHandler> Sortie de HandlerInt\n");

break;

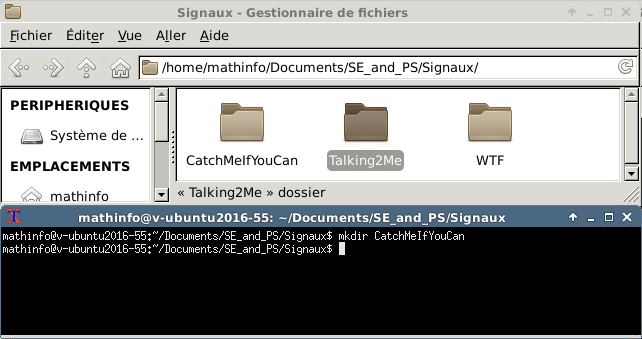
default:

printf("<signalHandler>Reception d'un signal inconnu\n");

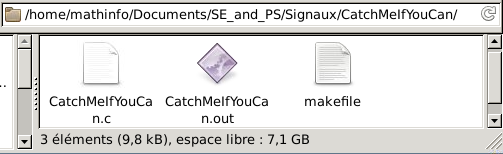
}

}V] Catch me if youcan

### Dans le repertoire "/home/mathinfo2014/Documents/SE\_and\_PS/Signaux", creez un repertoire "CatchMeIfYouCan".



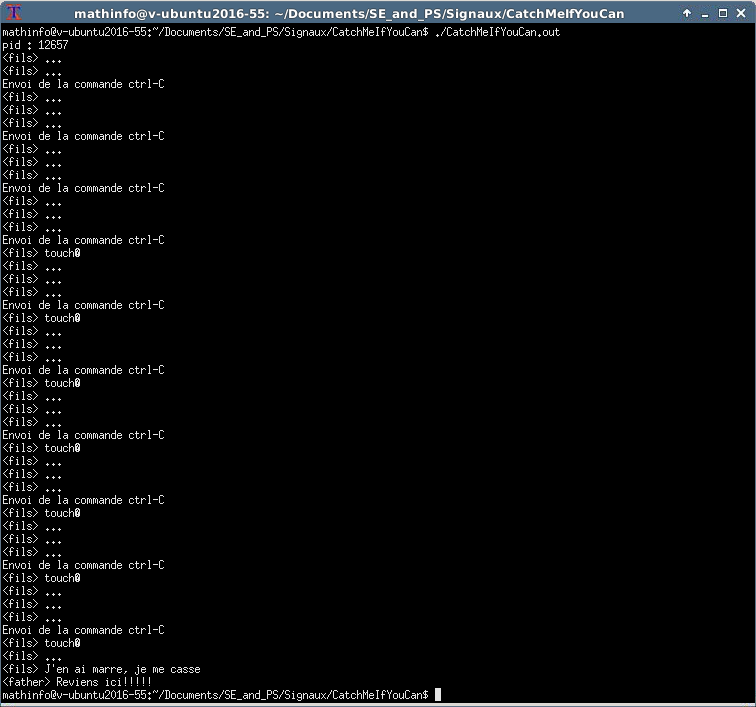
### Creez un fichier "catchMeIfYouCan.c" et un fichier "makefile".



### Corrigez le makefile pour compiler catchMeIfYouCan.c.

Z:\Documents\progra_system\1516_L2_SEPS_TDTP_4_STEGEN\Picture\e5.3.PNG

### Dans le fichier " catchMeIfYouCan.c", ecrivez un programme en C qui se scinde en: - un processus pere, qui intercepte le signal SIGCHLD indiquant qu’un de ses fils est termine, et affiche un message en tentant de generer des SIGINT toutes les 3 secondes a l’intention de son fils. Son callback ‘fatherSigHandler’ pour SIGCHLD acquitte son execution par un message et met fin a l’execution du processus pere.



### Code du programme

// includes habituel voir exercice précédant

voidfatherSigHandler (int);

voidsonSigHandler (int);

int main (){

int i;

structsigactionSigAction;

//variables utile du fork

if(child == -1)

// code d’erreur de création du processus

if(child == 0)

{

sigignore(SIGINT); // Désactive le signal

for(i=0;i<10;i++){

sleep(1);

printf("<fils> ...\n");

}

sigset(SIGINT); // Réactive le signal

SigAction.sa\_handler = sonSigHandler;

sigemptyset(&SigAction.sa\_mask);

SigAction.sa\_flags = 0;

if(sigaction(SIGINT,&SigAction,NULL) == -1)

perror("Erreur d'armement du signal SIGINT");

if(sigaction(SIGALRM,&SigAction,NULL) == -1)

perror("Erreur d'armement du signal SIGALRM");

alarm(20);

while(1){

sleep(1);

printf("<fils> ...\n");

}

}

else

{

printf("pid : %ld\n",getpid());

SigAction.sa\_handler = fatherSigHandler;

sigemptyset(&SigAction.sa\_mask);

SigAction.sa\_flags = 0;

if(sigaction(SIGCHLD,&SigAction,NULL) == -1)

perror("Erreur d'armement du signal SIGCHLD");

while(1){

sleep(3);

printf("<père>Envoi de la commande ctrl-C\n");

kill(child,SIGINT);

}

}

exit(0);

}

voidfatherSigHandler (intsignum){

switch(signum){

case SIGCHLD:

printf("<père> Reviens ici!!!!!\n");

exit(0);

break;

default:

printf("<signalHandler>Reception d'un signal inconnu\n");

}

}

voidsonSigHandler (intsignum){

switch(signum){

case SIGINT:

printf("<fils>touch%c\n",132);

break;

case SIGALRM:

printf("<fils> J'en ai marre, je me casse\n");

exit(0);

break;

default:

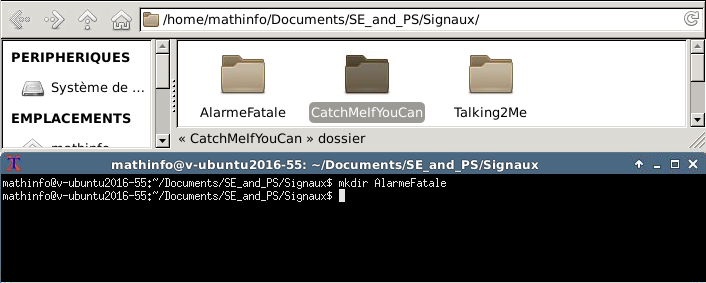
printf("<signalHandler>Reception d'un signal inconnu\n");

}

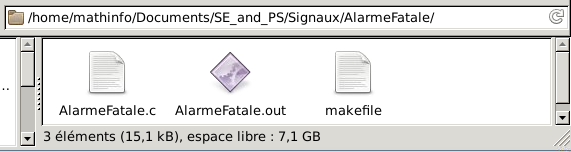
}

## VI] Alarme fatale

### Dans le repertoire "/home/mathinfo2014/Documents/SE\_and\_PS/Signaux", creez un repertoire "AlarmeFatale".



### Creez un fichier "alarmeFatale.c" et un fichier "makefile".



### Corrigez le makefile pour compiler alarmeFatale.c.

Z:\Documents\progra_system\1516_L2_SEPS_TDTP_4_STEGEN\Picture\e6.3.PNG

### Dans le fichier "alarmeFatale.c", ecrivez un programme en C qui intercepte le signal de l’alarme systeme (voir la fonction *alarm()*) SIGALRM. Le programme principal lancera une alarme prevue pour se declencher dans les deux prochaines secondes, et commencera a Ecrire un texte toutes les secondes pendant 5 secondes vous pouvez repeter la boucle d’affichage plusieurs fois eventuellement). Le callback de l’alarme nomme ‘alarmHandler’ affichera un message quand il est appelé.

### Z:\Documents\progra_system\1516_L2_SEPS_TDTP_4_STEGEN\Picture\e6.4.PNG

Dans ce programme, nous pouvons voir que quand le signal alarm(int) est lancé, Le programme va recevoir ce dernier après un certain temps demandé et réagir en écrivant « DRING, DRING, DRING » avanr de relancer le timer.

### Code du programme

//includes habituels voir exo précédant

voidAlarmeHandler (int);

int main (){

int i;

char tab[][50]={"belle,belle,belle","comme le jour","elle sont

toutes","belle,belle,belle","comme le jour"};

structsigactionSigAction;

// armement du signal SIGALRM

SigAction.sa\_handler = AlarmeHandler;

sigemptyset(&SigAction.sa\_mask);

SigAction.sa\_flags = 0;

if(sigaction(SIGALRM,&SigAction,NULL) == -1)

perror("Erreur d'armement du signal SIGALRM");

alarm(2);

for(i=0;i<5;i++){

sleep(1);

printf("%s\n",tab[i]);

}

exit(0);

}

voidAlarmeHandler (intsignum){

switch(signum){

case SIGALRM:

printf("<Alarme> DRING, DRING, DRING\n");

alarm(2);

break;

default:

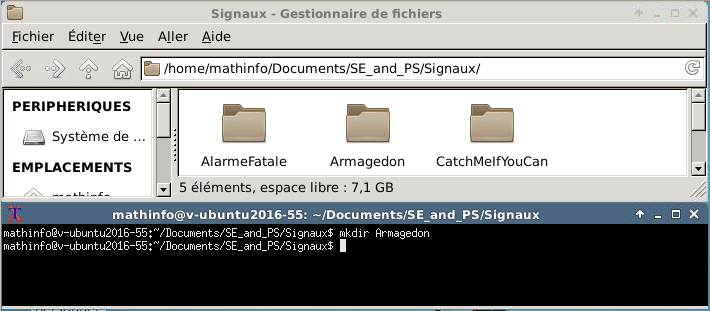
printf("<Alarme>Reception d'un signal inconnu\n");

}

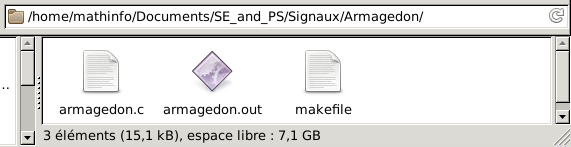
}

## VII] Armageddon

### Dans le répertoire "/home/mathinfo2014/Documents/SE\_and\_PS/Signaux", créez un répertoire "Armageddon".



### Créez un fichier "armagedon.c" et un fichier "makefile".



### Corrigez le makefile pour compiler armagedon.c.

Z:\Documents\progra_system\1516_L2_SEPS_TDTP_4_STEGEN\Picture\e7.3.PNG

### Dans le fichier "armagedon.c", écrivez un programme en C qui se scinde en un programme père et des programmes fils. Les quatre fils interceptent le signal dit ≪ utilisateur SIGUSR1 et affichent dans une boucle un message d’attente. Leur callback ‘sonSigHandler’ affiche un message puis provoque la levée du message SIGKILL pour lui-même. Le programme père affiche un message dans un boucle, et à la fin de la boucle déclenche le signal utilisateur pour tous ses fils et attends quelques secondes avant de terminer.

