# 1516 L2 SEPS TDTP 3 – Multi Proc Unix

## I] Connexion



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

## II] Un bon coup de fourchette

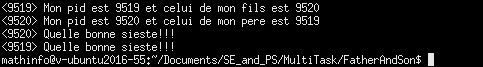
### Dans le répertoire "/home/mathinfo2014/Documents/SE\_and\_PS/MultiTask", créez un répertoire "FatherAndSon".

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

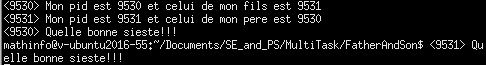


Le makefile devra contenir le texte suivant: fatherandson: fatherandson.c gcc -o fatherandson fatherandson.c -I.

### Dans le fichier "fatherandson.c", écrivez un programme en C qui se scinde en un processus père, qui affiche son PID et celui de son fils et patiente quelques secondes, et un processus fils qui affiche son PID et celui de son père et patiente également quelques secondes.



Ici, nous pouvons constater que le fils finit après le père.



Ici, le fils finit après le père mais le processus libère la console à la fin de ce dernier.

Z:\Documents\progra_system\TDTP3\Pictures\2.3.2.PNG

Ici nous pouvons constater que les PID se crée à la chaine (père 9438 et fils 9439 = père +1)

Le père a pour code :

printf("<%d> Mon pid est %d et celui de mon fils est

%d\n",getpid(),getpid(),child);

sleep(5);

printf("<%d> Quelle bonne sieste!!!\n",getpid());

exit(EXIT\_SUCCESS);

mais pour créer son fils :

if(child == -1)

{

perror("Fork error");

exit(EXIT\_FAILURE);

}

if(child == 0)

{

// code fils

}

else

{

// code père

}

Le fils a pour code :

printf("<%d> Mon pid est %d et celui de mon pere est

%d\n",getpid(),getpid(),getppid());

sleep(4);

printf("<%d> Quelle bonne sieste!!!\n",getpid());

\_exit(EXIT\_SUCCESS);

## III] Un jour c'est oui, un jour c'est non

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

Voir partie 1

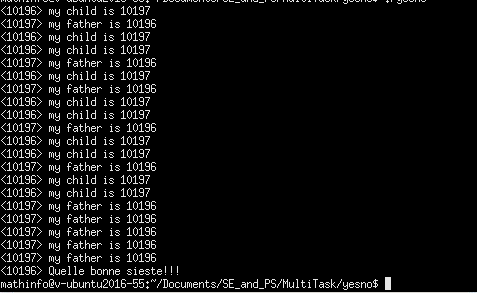
### Créez un fichier "yesno.c" et un fichier "makefile".



### Corrigez le makefile en conséquence.

Z:\Documents\progra_system\TDTP3\Pictures\3.2.1.PNG

### Dans le fichier " yesno.c", écrivez un programme en C qui se scinde en un processus père, qui affiche son PID et celui de son fils et patiente quelques secondes, le tout dans une boucle (de quelques itérations aussi), et un processus fils qui affiche son PID et celui de son père et patiente un temps légèrement supérieur, le tout aussi dans dans un boucle.



Le père a pour code :

for(i=0;i<it;++i)

{

printf("<%d> my child is %d\n",getpid(),child);

sleep(1);

}

printf("<%d> Quelle bonne sieste!!!\n",getpid());

exit(EXIT\_SUCCESS);

Il fait donc une boucle de quelques itération donnant son pid et celui de son fils créé.

Le fils a pour code :

for(i=0;i<it;++i)

{

printf("<%d> my father is %d\n",getpid(),getppid());

sleep(2);

}

\_exit(EXIT\_SUCCESS);

Il fait donc une boucle de quelques itération donnant son pid et celui de son père.

Sur cet exemple n’apparait pas un élément essentiel pour bien comprendre l’objectif de l’exercice.

Si le père finissait avant le fils, son PID ‘au fils) serait à 1.

### Comment faire pour que le processus père attende la fin du processus fils pour terminer ?

Il suffit simplement d’utiliser la fonction Wait(Null) qui permet d’attendre la fin d’un des processus lancé par le père.

Modifions le programme du père pour tenir compte de cela :

for(i=0;i<it;++i)

{

printf("<%d> my child is %d\n",getpid(),child);

sleep(1);

}

wait(NULL);

printf("<%d> Quelle bonne sieste!!!\n",getpid());

exit(EXIT\_SUCCESS);

## IV] Sweet Child O'Mine

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

Voir le point 1

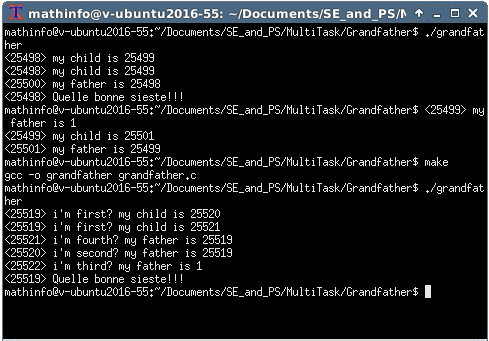
### Créez un fichier "grandfather.c" et un fichier "makefile".



### Corrigez le makefile en conséquence.

Z:\Documents\progra_system\TDTP3\Pictures\4.2.1.PNG

### Dans le fichier " grandfather.c", écrivez un programme en C qui se scinde en:



Dans ce programme, nous pouvons constater que les PID se suivent à la chaine mais que le programme éxécute directement le fils avant de rendre la main au père.

### Comment expliquez-vous les valeurs de PID observées ?

Vu qu’ils sont créé à la suite et qu’aucun autre programme n’utilise la même bande d’adresse que notre père les PID se suivent .

## V] O'brother

### Dans le répertoire "/home/mathinfo2014/Documents/SE\_and\_PS/MultiTask", créez un répertoire "GrandFatherDelayed"

Voir le point 1

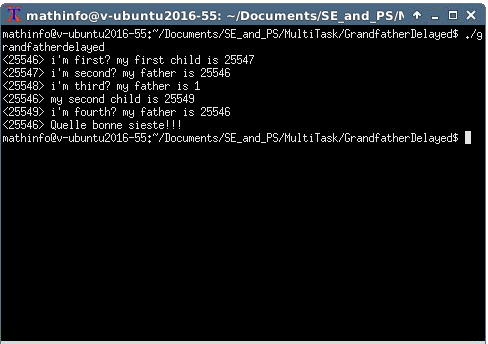
### Créez un fichier "grandfather.c" et un fichier "makefile".



### Corrigez le makefile en conséquence.

Z:\Documents\progra_system\TDTP3\Pictures\5.2.1.PNG

### Dans le fichier " grandfather.c", écrivez un programme en C.



Le père crée et exécute le premier fils qui a pour but de :

1. Dire qui il est.
2. Créé et éxécuter un fils.
3. Dire qu’il a fini.

Le premier fils et son fils ont pour code :

printf("<%d> i'm second? my father is %d\n",getpid(),getppid ());

child = fork();

if(child == -1) // Il n’a pas pu crée son fils

{

perror("second Fork error");

exit(EXIT\_FAILURE);

}

if(child == 0) // Il a créé son fils et le lance

{

printf("<%d> i'm third? my father is %d\n",getpid (),getppid());

\_exit(EXIT\_SUCCESS);

}

Else // Il fini son code quand il reprend la main

{

exit(EXIT\_SUCCESS);

}

Quand le père reprend la main,il dit qui il est et crée et exécute le deuxième fils qui a pour but de :

1. Dire qui il est.

Cette partie du code du père a pour code :

printf("<%d> i'm first? my first child is %d\n",getpid (),child);

sleep(5);

second = fork();

if(second == -1) // Il n’a pas pu crée son fils

{

perror("second Fork error");

exit(EXIT\_FAILURE);

}

if(second == 0) // Il a créé son 2ème fils et le lance

{

printf("<%d> i'm fourth? my father is %d\n",getpid (),getppid());

\_exit(EXIT\_SUCCESS);

}

Else // Il fini son code quand il reprend la main

{

printf("<%d> my second child is %d\n",getpid(),getppid ());

sleep(10);

printf("<%d> Quelle bonne sieste!!!\n",getpid());

exit(EXIT\_SUCCESS);

}

Dans cette version du 3, on a rajouter la fonction sleep() à chaque processus pour ralentir sa création ou faire en sorte que un fils se finisse après son père.

Comme nous pouvons le constater pour le troisème sur l’image précédente et le programme revient par défaut au PID 1 (init):

