USTHB **Année 2022/2023**
Faculté d'Informatique **SYST**
M1. SII

TP N°1
(Les processus)

Rappel 1: création de processus

Un processus est défini par son PID, et est créé par un processus père défini par un PPID. La fonction " ***int getpid()*** " permet de récupérer le PID du processus qui lôappel, et la fonction "***int getppid()***" permet de récupérer le PPID de son processus père

La fonction "***int fork()***" permet la création de processus sous Linux ; cette fonction retourne

-1 : en cas dôéchec

0: dans le processus fils créé

N : PID du processus fils nouvellement créé

```
# include <unistd.h>
# include <stdio.h>
Void main(){
intpid;
pid = fork ( ) ;
if (pid == - 1)
        { /* code si échec : printf ( ò le fork ( ) a échoué \n ò ) */}
if (pid == 0)
        { /* code correspondant à l'exécution du processus fils */ }
else
        { /* code correspondant à l'exécution du processus père */ }   }
```

Exercice 1 :

1- Ecrire un programme "C" qui crée deux processus (un père et son fils).

Le processus père affichera : "je suis le processus père, mon PID est:.... Et le PID de mon fils est : é ."

Le processus fils affichera : " je suis le processus fils, mon PID est :...Et le PID de mon père est :..."

2- Modifier le programme pour que le processus père lance deux processus fils.

Rappel 2 : La synchronisation entre processus : wait(), waitpid()

- Lôappel système ***wait(int *status)*** suspend lôexécution du processus appelant jusquôà ce quôun de ses fils se termine. Si un fils est déjà terminé, wait renvoie le PID du fils immédiatement sans bloquer. Il retourne l'identifiant du processus fils et son état de terminaison dans status (si status est différent de NULL). Si par contre le processus appelant ne possède aucun fils, wait retourne -1.
- Lôappel système ***waitpid(pid_t pid, int *status, int options)*** permet à un processus père d'attendre un fils particulier dôidentité **pid**, de façon **bloquante (options=0)** ou non bloquante **(options=WNOHANG)**

Si lôappel réussit, il renvoie l'identifiant du processus fils et son état de terminaison dans status (si status nôest pas NULL). En cas d'erreur -1 est renvoyé. Si lôoption WNOHANG est utilisée et aucun fils nôa changé dôétat, la valeur de retour est 0.

Exercice 2 :

Ecrire un programme "C" qui crée deux processus fils (un père et deux fils). Chaque processus fils affichera une valeur aléatoire. Réalisez les scénarios suivants :

Récupère la valeur aléatoire retournée par chaque processus fils et affichera la plus grande valeur.

Note :

Votre document devra avoir lêextension **.c**
Pour compiler : **gcc nom.c ï o nomcompilé**
Pour exécuter : **./nomcompilé**

Exercice 3

Ecrire un programme dont le père, après avoir créé trois fils(f1,f2,f3), attend le retour des trois fils pour réaliser le calcule x*y+z :

- Le fils f1 introduit une valeur x ;
- Le fils f2 introduit une valeur y ;
- Le fils f3 introduit une valeur z.

Exercice 4

En utilisant la fonction « execlp », écrire un programme qui permet dêexécuter la commande « ls ».

Exercice 5 à faire

Ecrire un programme carre.c qui affiche le carré dêune valeur saisie au lancement du programme

Ecrire un programme racine.c qui calcule la racine carré dêune valeur saisie au lancement du programme

Ecrire un programme calcule.c qui récupère une valeur et selon le choix de lêutilisateur (1 pour carré et 2 pour racine), crée un fils qui lancera le programme carre.c ou bien racine.c

Exercice 6 à faire

Modifier l'exercice2 pour créer N processus. N est introduit par l'utilisateur.

Remarques :

Le « main » dêun programme qui accepte des paramètres à son lancement	main(int argc, char *argv[])
Pour convertir vers un entier	val= atoi(caractere);
Pour convertir vers un caractère	sprintf(temp,"%d",val);
racine carrée	sqrt(val) ;
carré	pow(nombre, puissance);
Pour lancer un programme à partir dêun autre	execlp ("chemin","Nom",param, NULL);