PolytechLille
GIS2A4

TP de Système avancé
Exec – Dup - Pipe

Exercice 1 :
Ecrire, compiler et tester le programme suivant :

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
int main() {
        printf("lancement de la commande ls\n");
        if (execlp("ls", "ls", "-l", NULL) == -1) {
                perror("echec execlp");
                exit(1);
        }
        else {
                printf("fin de la commande ls\n");
                exit(0);
        }
        return 0 ;
}
```

Q1 : Justifier l'affichage obtenu
Q2 : Supprimer dans le code, tout ce qui est inutile.

Exercice 2 :
Ecrire le code d'un processus père qui crée un processus fils tels que :
- le processus père lit dans une variable fich, le nom d'un fichier à partir du clavier
- le processus fils exécute la commande wc -l fich, qui permet d'afficher le nombre de lignes du fichier fich.

Exercice 3 :
Soit les programmes somme.c et produit.c suivants :

```
/* fichier somme.c */
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
        int n1, n2;
        n1 = atoi(argv[1]); /* la fonction atoi convertit une chaîne de caractères en entier */
        n2 = atoi(argv[2]);
        printf("PID : %d, PPID : %d, %d + %d = %d\n", getpid(), getppid(), n1, n2, n1+n2);
        return 0 ;
}
```

```
/* fichier produit.c */
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
        int n1, n2;
        n1 = atoi(argv[1]); /* la fonction atoi convertit une chaîne de caractères en entier */
        n2 = atoi(argv[2]);
        printf("PID : %d, PPID : %d, %d * %d = %d\n", getpid(), getppid(), n1, n2, n1*n2);
        return 0 ;
}
```

Ces 2 programmes permettent respectivement de calculer la somme et le produit de 2 entiers passés en argument. Soit somme et produit les fichiers exécutables correspondants, testez l'exécution de ces 2 programmes en lançant par exemple les commandes :
somme 5 6
produit 4 8

Ecrire le code d'un processus sommeProduit qui prend de la même façon en argument 2 entiers et qui crée 2 processus fils concurrents. Le fils1 exécute le programme somme (avec les 2 entiers en arguments) et le fils2 exécute le programme produit (avec les 2 entiers en argument). Une fois la somme effectuée, le message "somme effectuée" doit être affiché. De même, une fois le produit effectué, le message "produit effectué" doit être affiché.

Exercice 4 :
Le but de cet exercice est de réaliser un programme permettant d'enchaîner des commandes.
Q1 : Réaliser un programme équivalent à l'enchaînement de commandes suivant : `who;ps;ls`. Ce programme doit fournir le même résultat que lorsqu'on lance cet enchaînement de façon interactive à partir d'un shell.
Avant toute réalisation, schématiser la hiérarchie de processus et les synchronisations à mettre en oeuvre (faire valider par l'enseignant AVANT la réalisation)
Q2 : Modifier le programme précédent pour qu'il enchaîne les commandes suivantes :
who ; ps -x ; ls -la

Exercice 5 :
Q1 : Réaliser un programme équivalent à la commande suivante : ls -la > dir.txt .

Q2 : Quelles sont les modifications à apporter au programme précédent pour qu'il réalise la commande ls -la >> dir.txt ?

Exercice 6 :
Ecrire le code C d'un processus père qui crée deux processus fils désignés par fils1 et fils2. Le processus fils1 lit une suite d'entiers terminée par 0 au clavier et transmet les entiers pairs au processus fils2 via un pipe. Le processus fils2 récupère les entiers pairs sur le pipe et affiche ceux qui sont supérieurs à un seuil S donné. En fin d'exécution des fils, le père affiche un message de terminaison.

Exercice 7 :
On souhaite réaliser un programme équivalent à la commande : cat /etc/passwd | wc .

Q1 : Proposer la hiérarchie de processus et les synchronisations à mettre en oeuvre.

Q2 : Réaliser le programme correspondant.