Unix TD2
Processus

Exercice 1 : Appels système fork , wait, waitpid.

Ecrire un programme C permettant de créer un processus fils affichant ses PID et PPID. Le père affichera ces mêmes informations.

Améliorer ce programme en permettant au père de créer un fils et d'afficher un message à la fin de l'exécution de ce fils.

Ajouter la possibilité de créer N fils. La valeur de N sera transmise en argument à l'exécution du fils.

Ajouter la possibilité pour le père d'afficher un message à la mort de chacun de ses fils. Le message devra donner le PID du fils décédé.

Ecrivez un programme C permettant d'observer (grâce à la commande ps) un zombi.

Exercice 2 : Appels sytème Exec

Ecrire un programme C permettant de lancer l'exécution d'une commande externe (ls, df, du, wc etc).

Améliorez ce programme en passant le nom du binaire en argument.

Ecrivez un programme permettant de créer un répertoire (commande « mkdir rep ») dont le nom sera passé en premier argument. Puis de créer dans ce répertoire un fichier dont le nom sera transmis en second argument.

Exercice 3 :

Ecire un programme C permettant de lancer l'exécution concurrente des binaires dont le nom est passé en argument (le nombre de binaires est quelconque). Affichez les pid et ppid des tâches et analysez les.

Exemple: $ prog1 cmd1 cmd2 cmd3 cmd4

Exercice 4 :

On considère le programme C (et le binaire exécutable correspondant) suivant:

```
/*      Exo4.c                  */
main()
{ int n=100;
printf("Bonjour > "); n*=2;
switch(fork())
{
        case 1: perror("Fork"); break;
        case 0: sleep(1); printf("dans le fils, adresse de n= %p\n",&n);
                n+=10; sleep(1);printf("n= %d\n",n);break;
        default: printf("dans le père, adresse de n= %p\n",&n); n+=10;
                sleep(3); printf("n= %d\n",n);
}
}
```

Qu'est ce qui est possible et qu'est ce qui ne l'est pas dans l'exécution suivante?

```
$ ./Exo4
Bonjour > dans le père, adresse de n= 142e8
Bonjour > dans le fils, adresse de n= 23200
n=210
n=220
```