TP 1 : Systèmes d'exploitation Avancés

EST beni mellal

Année : 2024/2025

ARI

Hachimi Youssef

Exercice 1 :

Le code en C de exercice 1:

GNU nano 4.8 ex1tp1.c

```c
#include <stdio.h> // input output
#include <stdlib.h> // pour fonction exet
#include <sys/types.h> // pour le type pid_t
#include <sys/wait.h> // por fonction wait();
#include <unistd.h> // pour gestion de processus aux linux


int main()
{
  pid_t pid = fork();
    if(pid == 0){
        printf("je suis le processus fils :%d \n",getpid());
    }
    else if(pid > 0){
        printf("je suis le processus  parent  :%d \n ",getppid());
    }
    else{
        printf("Erreur \n");
    }
return 0;
}
```

Compilation et Affichage de exercice 1:

```
root@ubuntu24:/home/ock# gcc ex1tp1.c -o ex1tp1.out
root@ubuntu24:/home/ock# ./ex1tp1.out
je suis le processus  parent  :16262
 je suis le processus fils :16658
root@ubuntu24:/home/ock#
```

Exerice 2 :

Le code en C de exercice 2 :

```
  GNU nano 4.8                                    ex2tp1.c
#include <stdio.h> // input / output
#include <sys/types.h> // pour le type pid_t
#include <sys/wait.h>// pour fonction wait
#include <stdlib.h>// pour fonction exit
#include <unistd.h> // pour gestion des processus
int main(){
        // declaration du variable pid
   pid_t pid = fork();
  // operation de fils
  if(pid == 0){
        printf("est l'operation de fils \n");
        int i;
        for(i = 1 ;i<= 10 ;i++){
         printf("%d\n",i);
         }
   }
  // operation de parent
  else if(pid > 0){
        wait(NULL);
        printf("est l'operation de parent \n");
        int i;
        for(i=11;i<=20 ;i++){
         printf("%d\n",i);
         }
   }
   // dans le cas de erreur
   else{
        printf("Erreur \n");
       }


return 0;


}
```

Compilation et Affichage de exercice 2:

```
root@ubuntu24:/home/ock# nano ex2tp1.c
root@ubuntu24:/home/ock# gcc ex2tp1.c -o ex2tp1
root@ubuntu24:/home/ock# ./ex2tp1
est l'operation de fils
1
2
3
4
5
6
7
8
9
10
est l'operation de parent
11
12
13
14
15
16
17
18
19
20
root@ubuntu24:/home/ock#
```

Exercice 3 :

Le code en C de exercice 3 :

home > ock > Documents > C test3.c > main()

```c
#include <stdio.h> // pour les fonctions input output come printf ,scanf .
#include <stdlib.h> // Inclure la bibliothèque standard pour des fonctions comme malloc .
#include <unistd.h>
#include <sys/types.h> // les définitions pour les types de données utilisés dans gestion des processus.
#include <sys/wait.h>  // pour fonction wait()



int main(){
    //declaration des variables type pid_t
    pid_t pid1 ,pid2 ,pid3 ,pid4 ,pid5 ,pid6;

    // pid1 = fork();
    // pid2 = fork();
    // pid3 = fork();
    // pid4 = fork();
    // pid5 = fork();
    // pid6 = fork();

    printf("le parent 1 est cree les fils : 2 , 3 , 4 \n");




    pid1 = fork();
    if(pid1 > 0){
        printf("je suis le parent 1 \n",getppid());
    }
    if(pid1 == 0){
        printf(" le fils %d de parent %d 1\n",getpid(),getppid());
    }
    else if(pid1 == -1){
        printf("Erreur de craetion \n");
```

```c
int main(){
    }
    else if(pid1 == -1){
        printf("Erreur de craetion \n");
        exit(1);
    }

    pid2 = fork();
    // if(pid1 > 0){
    //     printf("=== 1 ===\n");
    // }
    if(pid1 == 0){
        printf(" le fils %d de parent %d 1\n",getpid(),getppid());
    }
    else if(pid1 == -1){
        printf("Erreur de craetion \n");
        exit(1);
    }



    pid3 = fork();

    if(pid3 > 0){
        printf("=== 3 ===\n");
    }
    else if(pid3 == 0){
        printf("est le fils  de parent 1 \n");
    }
    else if(pid3 == -1){
        printf("Erreur de craetion \n");
    }
```

Compilation et Affichage de exercice 3:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

ock@ubuntu24:~$ gcc ex3tp1.c -o ex3tp1
ock@ubuntu24:~$ ./ex3tp1
1
3
4
2
6
5
ock@ubuntu24:~$
```

Exercice 4 :

Le code en C de exeerice 4 :

home > ock > C ex4tp1.c > main()

```c
#include <stdio.h> // input / output
#include <stdlib.h> // exit()
#include <unistd.h> // pour gestion des processus
#include <sys/types.h> // pour le type pid_t
#include <sys/wait.h> // les fonction comme wait()

#define NB_PROCESS 10 // le nomber de fils a cree
#define NB_ITERATION 10 // nomber de affichage de processus de fils

int main(){
    pid_t pid;
    int i,j;

    for(i=0 ; i < NB_PROCESS ; i++){
        pid = fork();
        if(pid < 0){
        printf("Erreur de creation du procesus fils\n");
        exit(1);
        }else if(pid == 0){
            // processus de fils
        for(j = 0; j < NB_ITERATION ;j++){
            printf("processus fils num %d :\n",i);
        }
        exit(0);
    }
    }
    for(i = 0;i < NB_PROCESS;i++){
        wait(NULL);
    }
    return 0;
}
```

Affichiage de exercice 4 :

0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2

3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5

6 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 8 8 8 8 8 8 8 8 8

9 9 9 9 9 9 9 9 9 9