



**Université Mohammed V – Rabat**

**École nationale supérieure d'informatique et d'analyse des systèmes**

## **COMPTE RENDU DU TP2 SYSTEM D'EXPLOITATION**

**Realiser Par : OTHMANE OUMOUDID**

## -EXO1

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <stdlib.h>
#include <wait.h>
#include <string.h>
#define N 10
#define NN N*sizeof(int)

int main(int argc , int **argv){

    int desc1[2],desc2[2];//desc[0] pour les lectures // desc[1] pour les ecritures
    int somme;
    int i,som;
    int *messageLire,*messageEcrire;//jouent le role de buffer
    messageLire=(int*)malloc(NN);
    messageEcrire=(int*)malloc(NN);
    pipe(desc1);
    pipe(desc2);
    pid_t pid=fork();

    if(pid==0){//processus fils
        close(desc1[1]);

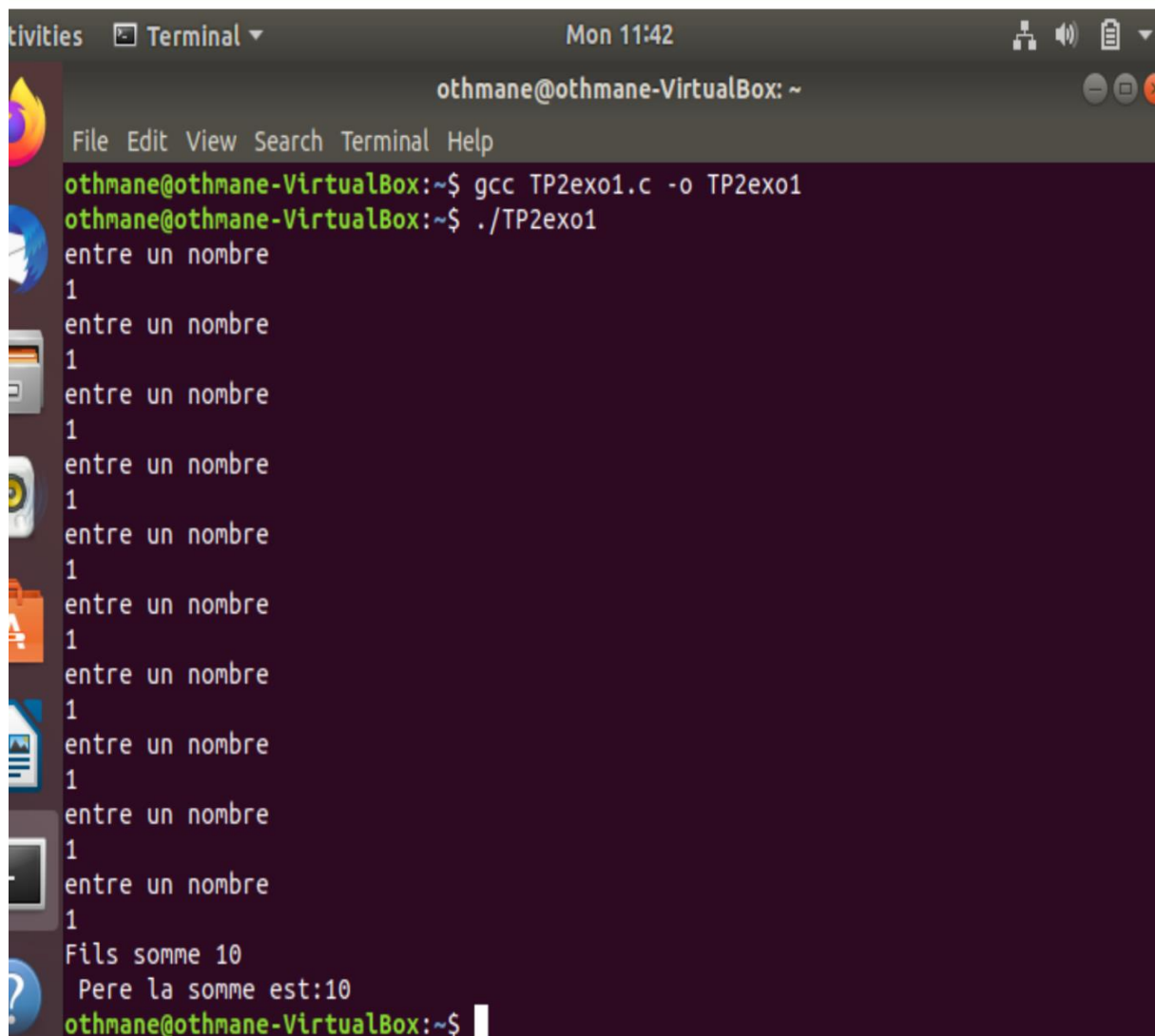
        read(desc1[0],messageLire,NN);
        somme=0;
        for(i=0;i<N;i++){
            somme=somme+messageLire[i];
        }
        printf("Fils somme %d\n",somme);
        close(desc2[0]);
        write(desc2[1],&somme,4);
        close(desc2[1]);close(desc1[0]);
        wait(NULL);
    }
    else {
        i=0;
        while(i<10){
            printf("entre un nombre\n");
            scanf("%d",&messageEcrire[i]);
            i++;
        }
        close(desc1[0]);//supprimer le droit de lecture au processus pere

        write(desc1[1],messageEcrire,NN);

        wait(NULL);

        close(desc2[1]);

        read(desc2[0],&som,4);
        close(desc2[0]);close(desc1[1]);
        printf(" Pere la somme est:%d\n",som);
    }
    return 0;}
```



```
othmane@othmane-VirtualBox: ~  
File Edit View Search Terminal Help  
othmane@othmane-VirtualBox:~$ gcc TP2exo1.c -o TP2exo1  
othmane@othmane-VirtualBox:~$ ./TP2exo1  
entre un nombre  
1  
entre un nombre  
1  
entre un nombre  
1  
entre un nombre  
1  
entre un nombre  
1  
entre un nombre  
1  
entre un nombre  
1  
entre un nombre  
1  
entre un nombre  
1  
entre un nombre  
1  
Fils somme 10  
Pere la somme est:10  
othmane@othmane-VirtualBox:~$
```

## -EXO2

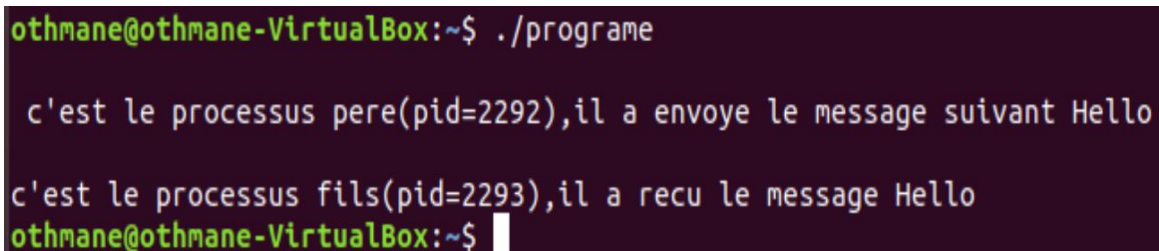
```
#include <stdio.h>  
#include <unistd.h>  
#include <sys/types.h>  
#include <stdlib.h>  
#include <wait.h>  
#include <string.h>
```

```
int main(int argc , int **argv){  
    int desc[2]; //desc[0] pour les lectures // desc[1] pour les ecritures  
    char messageLire[100], messageEcrire[100]; //jouent le role de buffer  
    pipe(desc);  
    pid_t pid=fork(); //return 0 au cas de processus fils et une valeur <> de zero au cas de proces-  
    sus pere  
    if(pid==0){ //processus fils  
        close(desc[1]); //supprimer le droit d'ecriture au processus fils  
        read(desc[0], messageLire, 100); //desc[0] transferer le message dans le buffer  
        printf("\n c'est le processus fils (pid=%d), il a reçu le message %s\n", getpid(), messageLire);  
    }  
}
```

```

else {
close(desc[0]);//supprimer le droit de lecture au processus pere
sprintf(messageEcrire,"Hello");//ecrire Hello dans messageEcrire
write(desc[1],messageEcrire,100);//ecrire dans desc[1] la chaine existe dans messageEcrire
printf("\n c'est le processus pere(pid=%d),il a envoye le message suivant %s\n",getpid(),messageEcrire);
wait(NULL);//pour que le processus pere attend que le processus pere fini d'ecrire
}
return 0;
}

```



```

othmane@othmane-VirtualBox:~$ ./programe

c'est le processus pere(pid=2292),il a envoye le message suivant Hello

c'est le processus fils(pid=2293),il a reçu le message Hello
othmane@othmane-VirtualBox:~$

```

## -EXO3

```

#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
#include <stdlib.h>
#include <wait.h>
#include <assert.h>

```

```

int main(int argc , char *argv[]){
pid_t pid_fils,pid;
int i;
int NP;
int Etat;
assert(argc == 2);//Pour verifie si on entre un argument ou Non
NP=atoi(argv[1]);
for(i=1 ; i<=NP ;i++){
pid=fork();
switch(pid){
case 0:
printf("Fils Num: %d, de pid :%d, Son pere pid est :%d\n",i,getpid(),getppid());
printf("\nJe vais attendre %d secondes avant fini\n",2*i);
sleep(2*i);
printf("\nLe fils %d va partir",i);

```

```

exit(i);
break;
case -1:
printf("\n on peut pas cree du processus fils\n");
exit(0);
break;
} }
i=0;
while(i<NP)
{ i++;
pid_fils = wait(&Etat);
if(pid_fils == -1){
printf("\nL'attent du Pere est fini");
exit(0);
}
else{
printf("\nPID DU FILS EST : %d\tEtat:%d\t Etat valeur :%d\n",pid_fils,Etat,WEXITSTA-
TUS(Etat));
}}
return 0;
}

```

```

othmane@othmane-VirtualBox:~$ gcc TP2exo3.c -o TP2exo3
othmane@othmane-VirtualBox:~$ ./TP2exo3 3
Fils Num: 1, de pid :2356, Son pere pid est :2355

Je vais attente 2 secondes avant fini
Fils Num: 2, de pid :2357, Son pere pid est :2355

Je vais attente 4 secondes avant fini
Fils Num: 3, de pid :2358, Son pere pid est :2355

Je vais attente 6 secondes avant fini

Le fils 1 va partir
'est le Pere---PRFils---PID : 2356      Etat:256      Etat valeur :1

Le fils 2 va partir
'est le Pere---PRFils---PID : 2357      Etat:512      Etat valeur :2

Le fils 3 va partir
'est le Pere---PRFils---PID : 2358      Etat:768      Etat valeur :3
othmane@othmane-VirtualBox:~$ █

```

Act  
Acc

## -EXO4

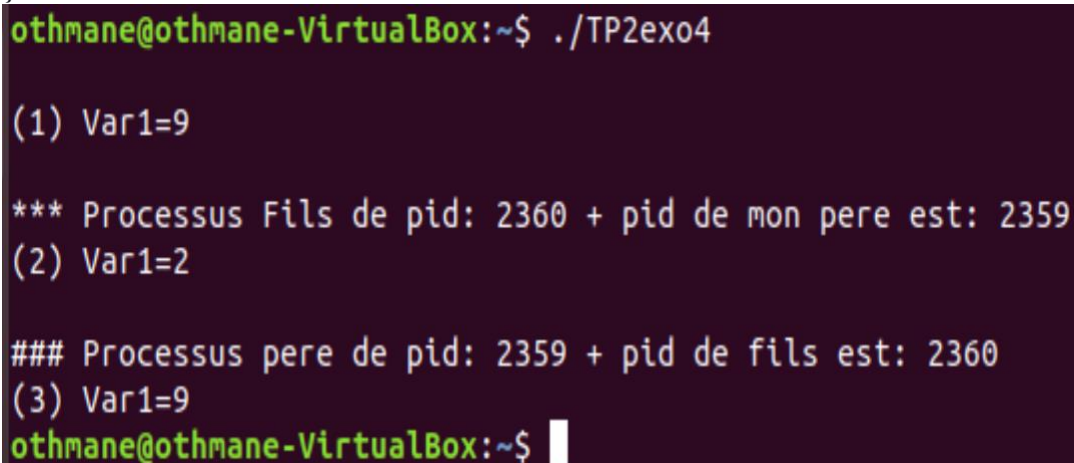
```

#include <stdio.h>
#include <unistd.h>

```

```
#include <wait.h>
```

```
int main(){
    pid_t pid;
    int var1=9;
    printf("\n(1) Var1=%d\n",var1);
    pid=fork();
    if(pid==-1)
        printf("\n Y'a pas de possibilite de cree une processus fils");
    else if(pid==0){
        printf("\n*** Processus Fils de pid: %d + pid de mon pere est: %d",
            (int) getpid(),(int) getppid());
        var1=2;
        printf("\n(2) Var1=%d\n",var1);
    }
    else {
        wait(NULL);
        printf("\n### Processus pere de pid: %d + pid de fils est: %d",
            (int) getpid(),(int) pid);
        printf("\n(3) Var1=%d\n",var1);
    }
    return 0;
}
```



```
othmane@othmane-VirtualBox:~$ ./TP2exo4

(1) Var1=9

*** Processus Fils de pid: 2360 + pid de mon pere est: 2359
(2) Var1=2

### Processus pere de pid: 2359 + pid de fils est: 2360
(3) Var1=9
othmane@othmane-VirtualBox:~$
```

```
#include <stdio.h>
```

```
#include <unistd.h>
```

```
#include <pthread.h>
```

```
int main(){
    pthread_t thread_id;
    int val=10;
    int i;
    printf("(1) val=%d\n",val);
    void* fonction_thread(void* parametre){
        val=2;
        printf("\n***Fonction thread de thread_id:%d\n",
            (int) pthread_self());
        printf("\n (3)val=%d\n",val);
    }
```

```

return NULL;
}
printf("\n###Dans le main , thread_id:%d\n",(int)pthread_self());
printf("\n(2) val=%d\n",val);

```

```

pthread_create (&thread_id,NULL,&fonction_thread,NULL);

```

```

for(i=0;i<10000;i++);//pour que le thread trouve le temps a s'executer , on peut aussi utiliser
pthread_join(thread_id,NULL);

```

```

printf("\n!!!!Dans le main , thread_id:%d\n",(int)pthread_self());
printf("\n(4) val=%d\n",val);
return 0;
}

```

```

othmane@othmane-VirtualBox:~$ ./TP2
(1) val=10

###Dans le main , thread_id:1358391104

(2) val=10

***Fonction thread de thread_id:1349871360

(3)val=2

!!!!Dans le main , thread_id:1358391104

(4) val=2
othmane@othmane-VirtualBox:~$

```

## -EXO5

```

#include <stdio.h>
#include <unistd.h>
#include <wait.h>

```

```

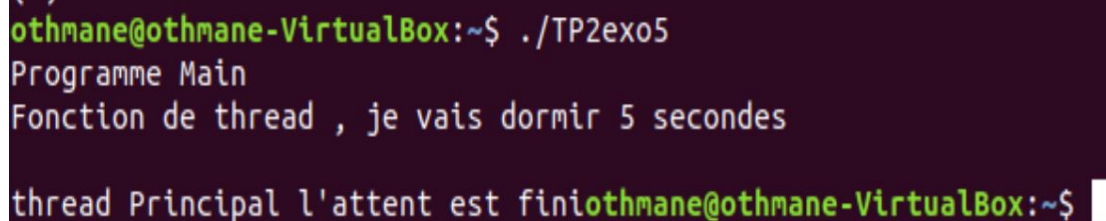
int main(){
pid_t pid;
int var1=9;
printf("\n(1) Var1=%d\n",var1);
pid=fork();
if(pid==-1)
printf("\n Y'a pas de possibilite de cree une processus fils");

```

```

else if(pid==0){
printf("\n*** Processus Fils de pid: %d + pid de mon pere est:
%d", (int) getpid(), (int) getppid());
var1=2;
printf("\n(2) Var1=%d\n", var1);
}
else {
wait(NULL);
printf("\n### Processus pere de pid: %d + pid de fils est: %d", (int) getpid(), (int) pid);
printf("\n(3) Var1=%d\n", var1);
}
return 0;
}

```



```

othmane@othmane-VirtualBox:~$ ./TP2exo5
Programme Main
Fonction de thread , je vais dormir 5 secondes

thread Principal l'attent est fini
othmane@othmane-VirtualBox:~$

```

## -EXO6

```

#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
#include <stdlib.h>
#include <wait.h>
#include <pthread.h>
#define N 10

```

```

void* fonction_thread(void* tab){
int T=(int)tab;
int j;
printf("\n\t\tLa fonction de thread\n");
for(j=0;j<N;j++){
printf("\n\t\tl'element %d est %d\n",j,T[j]);
}
}

```

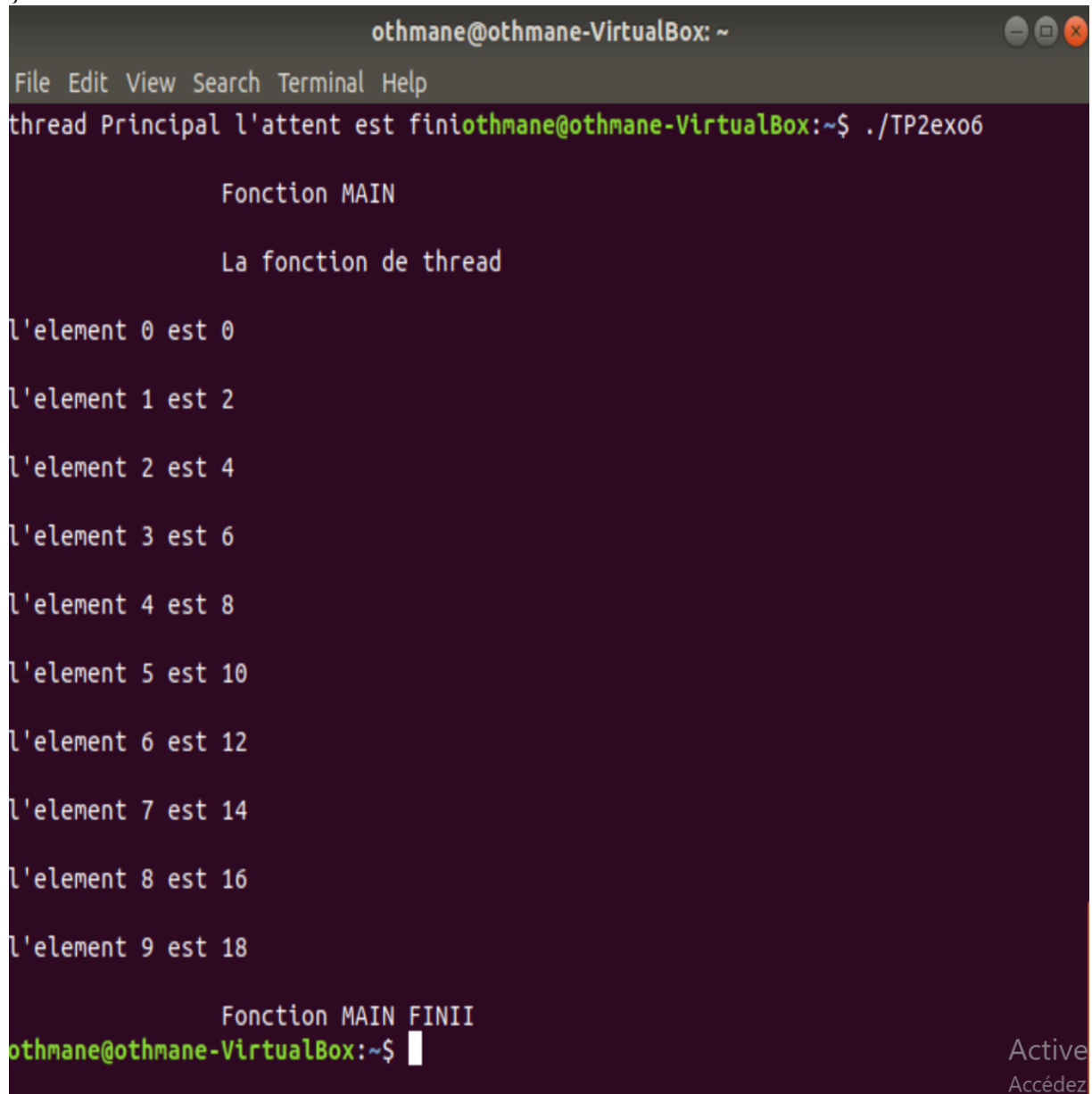
```

int main(){
pthread_t thread_id;
int *tab;
int i;
tab=(int*)malloc(N*sizeof(int));
printf("\n\t\tFonction MAIN\n");
for(i=0;i<N;i++){
tab[i]=2*i;
}
}

```



```
pthread_create(&thread_id,NULL,&fonction_thread,tab);
pthread_join(thread_id,NULL);
printf("\n\tFonction MAIN FINII\n");
return 0;
}
```



```
othmane@othmane-VirtualBox: ~
File Edit View Search Terminal Help
thread Principal l'attent est finiothmane@othmane-VirtualBox:~$ ./TP2exo6

    Fonction MAIN

    La fonction de thread

l'element 0 est 0
l'element 1 est 2
l'element 2 est 4
l'element 3 est 6
l'element 4 est 8
l'element 5 est 10
l'element 6 est 12
l'element 7 est 14
l'element 8 est 16
l'element 9 est 18

    Fonction MAIN FINII
othmane@othmane-VirtualBox:~$
```

Active  
Accédez



