



Solution: Practical work (TP) test (15 points)

Exercise n° 1 (5 points)

What is the result displayed on the screen when each of the following two programs is run?

Program 1

```
int i=4,j=10;
pid_t p;
int main()
{
    p = fork();
    if(p<0) exit(1);
    j += 2;
    if (p == 0){
        i *= 3;
        j *= 3;
    }
    else {
        i *= 2;
        j *= 2;
    }
    printf("i=%d, j=%d \n", i,j);
    return 0;
}
```

Program 2

```
int main()
{
    int i;
    for (i = 0; i < 4; i++)
        if (fork () == 0)
        {
            printf ("A\n");
            exit(0);
        }
    printf ("B\n");
    while (wait(NULL) >= 0);
}
```

Program 1: (2.5 pts)

```
mohamed@mohamed-VirtualBox:~/test2024$ ./exo8
i=8 , j=24
i=12 , j=36
```

Program 2: (2.5 pts)

```
mohamed@mohamed-VirtualBox:~/test2024$ ./exo9
A
B
A
A
A
mohamed@mohamed-VirtualBox:~/test2024$ ./exo9
B
A
A
A
A
```

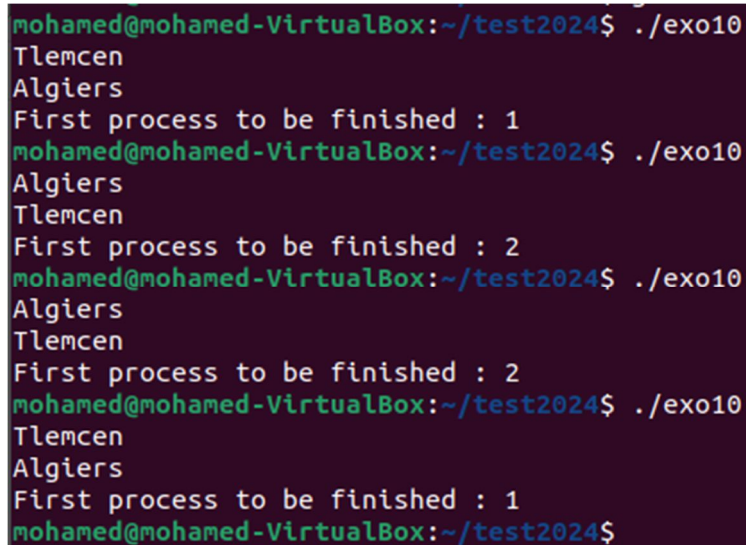
Peu importe l'ordre d'affichage, l'essentiel l'étudiant doit mentionner qu'il y a l'affichage de quatre "A" par le premier fils et ses descendants et un seul "B" par le père (Affichage de 4 "A" et 1 "B").

Exercise n° 2 (5 points)

Write a program that creates 2 processes, one displaying "*Tlemcen*" and the other "*Algiers*". The parent will have to wait for its two children to finish and display which was the first process to finish.

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

int main() {
    pid_t pid1, pid2, pid_first;
    int status;
    switch(pid1=fork()) {
        case -1: perror("fork error"); break;
        case 0: printf("Tlemcen\n"); exit(0); break;
        default:
            switch(pid2=fork()) {
                case -1: perror("fork error"); break;
                case 0: printf("Algiers\n"); exit(1); break;
                default: break;
            }
            break;
    }
    pid_first = wait(&status); // Permet de récupérer le pid du premier
                                // fils qui vient de se terminer
    wait(&status); // Pour le deuxième fils
    printf("First process to be finished : %d\n", (pid_first==pid1)?1:2);
}
```



```
mohamed@mohamed-VirtualBox:~/test2024$ ./exo10
Tlemcen
Algiers
First process to be finished : 1
mohamed@mohamed-VirtualBox:~/test2024$ ./exo10
Algiers
Tlemcen
First process to be finished : 2
mohamed@mohamed-VirtualBox:~/test2024$ ./exo10
Algiers
Tlemcen
First process to be finished : 2
mohamed@mohamed-VirtualBox:~/test2024$ ./exo10
Tlemcen
Algiers
First process to be finished : 1
mohamed@mohamed-VirtualBox:~/test2024$
```

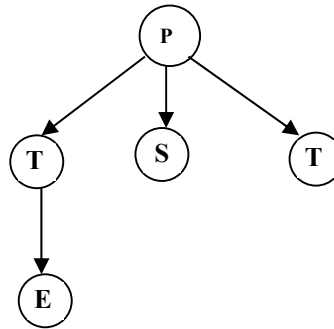
La création des processus et l'affichage de chacun d'eux : (2.5 points)

Les trois dernières instructions : (2.5 points)

- La synchronisation en utilisant les `wait()` en particulier le premier `wait(&status)` qui récupère le `pid` du premier processus qui vient de se terminer,
- L'affichage du test contenu dans "printf".

Exercise n° 3 (5 points)

1) Write a program in C language to create the following tree structure:



2) Using the functions *wait(0)* and *exit(0)*, synchronize the different processes to display: TEST.

Je propose deux versions : une comme celle du TD n°2 et une autre avec switch.

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

int main() {
    pid_t pid1,pid2,pid11,pid3;
    int status;

    switch(pid1=fork()) {    // Child 1
    case -1: perror("fork error"); break;
    case 0:
        printf("T\n");
        switch(pid11=fork()) {    // Child 1.1
        case -1: perror("fork error"); break;
        case 0:  printf("E\n"); exit(0);break;
        default : wait(NULL); exit(0); break;
        }
    default: wait(NULL);
        switch(pid2=fork()) {    // Child 2
        case -1: perror("fork error"); break;
        case 0:
            printf("S\n"); exit(1); break;
        default: wait(NULL);
            switch(pid3=fork()) {    // Child 3
            case -1: perror("fork error"); break;
            case 0: printf("T\n"); exit(1); break;
            default: wait(NULL); break;
            }
        }
    }
}
```

```

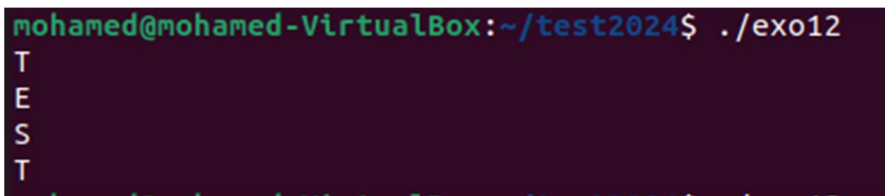
#include<sys/types.h>
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
#include<sys/wait.h>

int main(void) {
if (fork()==0) { //child 1
    printf("T");
    fflush(stdout);
    if(fork()==0){ //child 1.1
        printf("E");
        //fflush(stdout);
        exit(0);
    }
    wait(NULL);
    exit(0);
}
else{
    wait(NULL);
    if (fork()==0) { //child 2
        printf("S");
        fflush(stdout);
        exit(0);
    }
    else{
        wait(NULL);
        if (fork()==0) { //child 3
            printf("T");
            fflush(stdout);
            exit(0);
        }
        wait(NULL);
        exit(0);
    }
}
}
}

```

Question (1) : La structure du programme et l’affichage sans synchronisation (3 pts)

Question (2) : Synchronisation entre les processus en utilisant *wait()* et *exit()* ainsi que leurs emplacements dans le programme. (2 pts)



```

mohamed@mohamed-VirtualBox:~/test2024$ ./exo12
T
E
S
T

```

Seulement dans cet exemple j’ai fait un retour à la ligne après chaque affichage pour bien illustrer le résultat de l’affichage.