TP n°2: Process management (2)

Exercise n°1

1) Write a program that creates N child processes in parallel, then waits for them to terminate. Each process created must display its pid, the pid of its parent and its creation order number.
2) In this program, each of the children must return its creation number to the termination. The main process must display the output value of each of the children and the pid of the child whose termination it has just detected.

Exercise n°2

Write a program using the functions *fork()*, *exit()* and *wait()* to transmit an ASCII character from the child process to the parent process. The child process reads the character using the function *getchar()*. The parent process displays the ASCII code of this character and transforms it into uppercase using the function *toupper(char c)*.

Exercise n°3

Use the compiler "c" under Linux "gcc" to compile the following program.

#gcc -c filename.c

#gcc -o exe-name filename.o

#./exe-name

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

int main(void){
int x,pid,s;

pid =fork();
if (pid){
          wait(&x);
          s = x>>8;
          s = s*s;
          printf("The parent process calculates the square of x :%d\n",s);
          exit(0);
          }
else {
       printf("entrer la valeur de x : \n");
       scanf("%d",&x);
       exit(x);
      }
return 0;
}
```

1) Indicate the result displayed by this program.
2) Indicate the relation between *exit()* and *wait()*.

Exercise n° 4

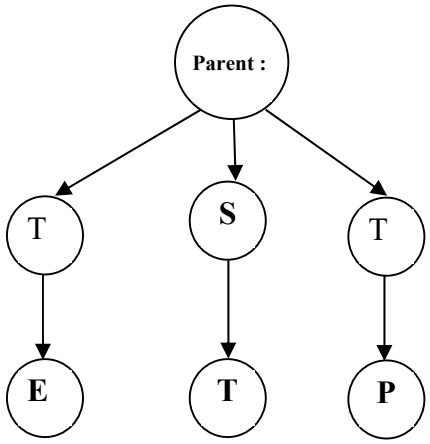
Write a program whose parent, after creating three children (f1, f2, f3), waits for these three children to return before performing the calculation 3 × 10 + 5.

The data:

- The child f1 returns the value 5;
- The child f2 returns the value 10;
- The child f3 returns the value 3.

Exercise n° 5

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



2) Synchronize the different processes to display: Parent: TEST TP using the wait(0) and exit(0) functions.