TP N°: 1 : Correction

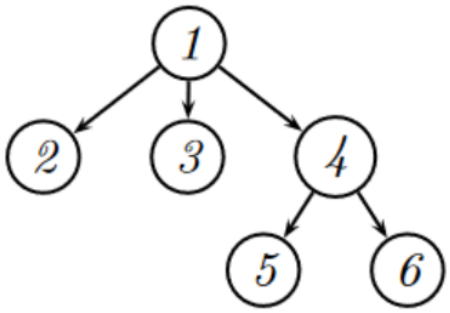
Exercice 1

Ecrivez un programme en C qui utilise la fonction fork() pour créer un processus fils. Le processus fils doit afficher les nombres de 1 à 10, tandis que le processus parent doit afficher les nombres de 11 à 20.

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

int main() {
    pid_t pid = fork();

    if (pid == -1) {
        printf("Erreur lors de la création du processus fils");
        exit(1);
    }

    if (pid == 0) {
        // Code du processus fils
        for (int i = 1; i <= 10; i++) {
            printf("Processus fils: %d \n", i);
        }
    } else {
        // Code du processus parent
        wait(NULL);
        for (int i = 11; i <= 20; i++) {
            printf("Processus parent: %d \n", i);
        }
    }

    return 0;
}
```

Exercice 2

Ecrire un programme qui engendre l'arbre généalogique suivant :



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

int main() {
    printf("1 \n");
    pid_t pid2 = fork();

    if (pid2 == -1) {
       printf("Erreur lors de la création du processus fils");
        exit(1);
    } else if (pid2 == 0) {
        printf("2 \n");
    } else {
        pid_t pid3 = fork();

        if (pid3 == -1) {
            printf("Erreur lors de la création du processus fils");
            exit(1);
        } else if (pid3 == 0) {
            printf("3 \n");
        } else {
            pid_t pid4 = fork();

            if (pid4 == -1) {
                printf("Erreur lors de la création du processus fils");
                exit(1);
            } else if (pid4 == 0) {
                printf("4 \n");
                pid_t pid5 = fork();

                if (pid5 == -1) {
                   printf("Erreur lors de la création du processus fils");
                    exit(1);
                } else if (pid5 == 0) {
                    printf("5 \n");
                } else {
                    pid_t pid6 = fork();

                    if (pid6 == -1) {
                        printf("Erreur lors de la création du processus fils");
                        exit(1);
                    }
                    if (pid6 == 0) {
                        printf("6 \n");
                    }
                }
            }
        }
    }

    return 0;
}
```

Exercice 3

Ecrivez un programme en C qui crée un processus fils à l'aide de la fonction fork(). Le processus fils doit afficher son identifiant de processus (PID) ainsi que l'identifiant de processus de son parent (PPID). Le processus père doit attendre la fin de l'exécution du processus fils en utilisant la fonction wait().

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

void main() {
    pid_t pid;
    pid = fork();
    if (pid < 0) {
        printf("Erreur lors de la création du processus fils \n");
        exit(1);
    } else if (pid == 0) {
        // Processus fils
        printf("Je suis le processus fils (PID=%d) de parent (PPID=%d) \n", getpid(), getppid());
        exit(0);
    } else {
        // Processus père
        printf("Je suis le processus père (PID=%d) de fils (PID=%d)\n", getpid(), pid);
        wait(NULL);
        printf("Le processus fils a terminé \n");
    }
}
```

Exercice 4

Écrire un programme en C qui crée 10 processus fils. Chacun d'entre eux devra afficher dix fois d'affilée son numéro d'ordre entre 0 et 9.

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

#define NB_PROCESS 10
#define NB_ITERATIONS 10
void main() {
    pid_t pid;
    int i, j;

    for (i = 0; i < NB_PROCESS; i++) {
        pid = fork();
        if (pid < 0) {
            printf("Erreur lors de la création du processus fils\n");
            exit(1);
        } else if (pid == 0) {
            // Processus fils
            for (j = 0; j < NB_ITERATIONS; j++) {
                printf("%d ", i);
            }
            exit(0);
        }
    }
    // Processus père
    for (i = 0; i < NB_PROCESS; i++) {
        wait(NULL);
    }
}
```