

## TP2. Memory Management

### A- Shared Memory

1. Dans ce programme, on incrémente dans le processus fils la valeur de `i` et la valeur de `*ptr`. Dans la sortie du programme, on remarque que l'incréméntation du pointeur `*ptr` affecte les deux process. Dans l'enfant et dans le parent.  
L'incréméntation de la valeur de `i` quant à elle n'intervient que dans le processus fils.  
On peut supposer que `*ptr` est une donnée partagée tandis que `i` ne l'est pas.

```
0x00000000 3473432  adrito  600  4194304  2

Value of *ptr = 55
Value of i = 55
Value of *ptr = 55
Value of i = 54
adrito@ubuntu:~/Documents/ECE/Lab2$
```

2.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/shm.h>
#include <sys/wait.h>
#include <unistd.h>
#define KEY 4567
#define PERMS 0660

int main(int argc, char **argv) {
    int id; // Déclaration de la variable integer "id"
    int i; // Déclaration de la variable integer "i"
    int *ptr; // Déclaration d'un pointeur de type integer

    system("lpc -n"); // Récupération et affichage d'informations sur les segments de mémoire partagés actifs.
    id = shmget(KEY, sizeof(int), IPC_CREAT | PERMS); // id prend l'identifiant du segment de mémoire partagée associé à la valeur de l'argument KEY. On alloue un segment pour nos 2 process.
    system("lpc -n"); // Récupération d'informations sur les segments de mémoire partagés actifs après la création.

    ptr = (int *) shmat(id, NULL, 0); // L'adresse du pointeur (calling process) est attachée au nouveau segment de mémoire partagée.
    *ptr = 54; // ptr prend la valeur 54. La mémoire partagée prends la valeur 54.
    i = 54; // i prend la valeur 54 de manière locale.

    if (fork() == 0) {
        (*ptr)++; // Incréméntation de la valeur du pointeur (partagée)
        i++; // Incréméntation de i en local
        printf("Value of *ptr = %d\nValue of i = %d\n", *ptr, i); // Affichage de la valeur des variables.
        exit(0); // On quitte la boucle.
    } else {
        wait(NULL); // On attend le changement d'état du fils du processus appelant
        printf("Value of *ptr = %d\nValue of i = %d\n", *ptr, i); // Affichage de la valeur des variables.
        shmctl(id, IPC_RMID, NULL); // Exécute l'opération de contrôle IPC_RMID (marquer le segment pour destruction) sur le segment "id". En gros libération "free" de l'espace utilisé.
    }
}
```

3.

```
int main()
{
    int id;
    int *ptr;

    id = shmget(KEY, sizeof(int), IPC_CREAT | PERMS);
    ptr = (int *) shmat(id, NULL, 0);

    if(fork()==0)
    {
        int a = 2;
        int b = 3;
        *ptr = (a+b);
        exit(0);
    }
    else
    {
        wait(NULL);
        int c = 1;
        int d = 2;
        *ptr -= (c+d);
        printf("(a+b)-(c+d) = %d\n", *ptr);
    }
    shmctl(id, IPC_RMID, NULL);
    return 0;
}
```

## B- Parallel Computing

### 1- Methode avec wait(NULL)

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/shm.h>
#include <sys/wait.h>
#include <unistd.h>
#define KEY1 4567
#define KEY2 4568
#define PERMS 0660

int main()
{
    int id1, id2, *ptr1, *ptr2;
    int a=1, b=1, c=4, d=2, e=1, f=1;

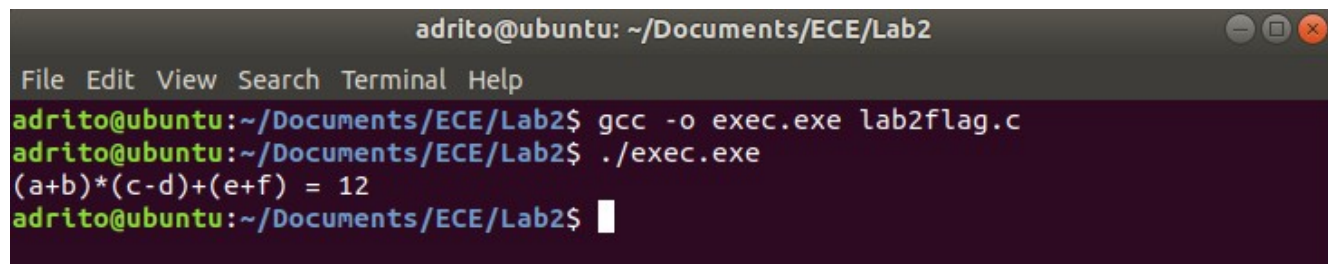
    id1 = shmget(KEY1, sizeof(int), IPC_CREAT | PERMS);
    id2 = shmget(KEY2, sizeof(int), IPC_CREAT | PERMS);
    ptr1 = (int *) shmat(id1, NULL, 0);
    ptr2 = (int *) shmat(id2, NULL, 0);

    //1er fils
    if(fork()==0)
    {
        *ptr1 = (a+b);
        exit(0);
    }

    //parent
    else
    {
        int res = (c-d);
        wait(NULL);
        res = res * (*ptr1);
        //2e fils
        if(fork()==0)
        {
            *ptr2 = (e+f);
            exit(0);
        }
        //parent
        else
        {
            wait(NULL);
            res = res + (*ptr2);
            printf("(a+b)*(c-d)+(e+f) = %d\n", res);

            shmctl(id1, IPC_RMID, NULL);
            shmctl(id2, IPC_RMID, NULL);
            return 0;
        }
    }
}
```

Méthode avec les flags :



```
adrito@ubuntu: ~/Documents/ECE/Lab2
File Edit View Search Terminal Help
adrito@ubuntu:~/Documents/ECE/Lab2$ gcc -o exec.exe lab2flag.c
adrito@ubuntu:~/Documents/ECE/Lab2$ ./exec.exe
(a+b)*(c-d)+(e+f) = 12
adrito@ubuntu:~/Documents/ECE/Lab2$
```

The image shows a terminal window titled "adrito@ubuntu: ~/Documents/ECE/Lab2". The window has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal content shows the user compiling a C file "lab2flag.c" into an executable "exec.exe" using the command "gcc -o exec.exe lab2flag.c". Then, the user runs the executable with the command "./exec.exe", which outputs the result of a mathematical expression: "(a+b)\*(c-d)+(e+f) = 12". The prompt "adrito@ubuntu:~/Documents/ECE/Lab2\$" is visible at the end of the line.

```

lab2flag.c
~/Documents/ECE/Lab2
Open Save

#include <sys/shm.h>
#include <sys/wait.h>
#include <unistd.h>

#define KEY1 4567
#define KEY2 4568
#define KEY3 4569
#define PERMS 0660

int main()
{
    int id1, id2, id3, *ptr1, *ptr2, *flag;
    int a=3, b=2, c=6, d=4, e=1, f=1;

    id1 = shmget(KEY1, sizeof(int), IPC_CREAT | PERMS);
    id2 = shmget(KEY2, sizeof(int), IPC_CREAT | PERMS);
    id3 = shmget(KEY3, sizeof(int), IPC_CREAT | PERMS);
    ptr1 = (int *) shmat(id1, NULL, 0);
    ptr2 = (int *) shmat(id2, NULL, 0);
    flag = (int *) shmat(id3, NULL, 0);

    //1er fils
    if(fork()==0)
    {
        *ptr1 = (a+b);
        *flag = 1;
        exit(0);
    }

    //parent
    else
    {
        int res = (c-d);
        while(*flag != 1){}
        res = res * (*ptr1);
        //2e fils
        if(fork()==0)
        {
            *ptr2 = (e+f);
            *flag = 2;
            exit(0);
        }
        //parent
        else
        {
            while(*flag != 2){}
            res = res + (*ptr2);
            printf("(a+b)*(c-d)+(e+f) = %d\n", res);
            shmctl(id1, IPC_RMID, NULL);
            shmctl(id2, IPC_RMID, NULL);
            return 0;
        }
    }
}

```

Avec la methode des flags, on obtient le même résultat. La différence réside dans le fait que l'on a remplacé le wait par une méthode manuelle à savoir les flags.