Lab2: Programming lab, memory management

Subject

- Create a new file : `touch myfile.c`
- Editing a file : `nano myfile.c`
- Running a file : `gcc -o execName myfile.c` then `./execName` to display results.

Shared Memory

1- What could you inner from the output regarding the state of `i` and `*ptr` ?

We could infer from the state of `i` and `*ptr` that the value of `i` isn't shared between a child and his parents, however the value of `*ptr` is.

2- Read the code carefully and add your comments to all the lines

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

int main(int argc, char **argv)
{
    int id;
    int i;
    int *ptr;
    system("ipcs -m"); //Show stats of IPCs
    id = shmget(KEY, sizeof(int), IPC_CREAT | PERMS); //Create an IPC of 4 bytes (=sizeof int) with shared memory (=allocate shared
    system("ipcs -m"); //Show again stats of IPCs because we just created another
    ptr = (int *) shmat(id, NULL, 0); //Attach an existing shared memory (id) to an adress space (ptr)
    *ptr = 54;i = 54;
    if (fork() == 0) //Creating a child process
    {//Parent
        (*ptr)++;i++; //Increment *ptr and i in the parent
        printf("Value of *ptr = %d\nValue of i = %d\n", *ptr, i); //Print the value of *ptr and i
        exit(0); //Close the parent process
    }
    else
    { //Child
        wait(NULL); //Wait until the parent process finish
        printf("Value of *ptr = %d\nValue of i = %d\n", *ptr, i); //Print the value of *ptr and i
        shmctl(id, IPC_RMID, NULL); //Function of control of the memory
    }
}
```

3- Write a program that computes the following expression `((a+b)-(c+d))` using a parent and a child process.

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

int main(int argc, char **argv)
{
    int id, id2;
    int *ptr, *ptr2;
    int total;
    int a = 1, b = 2, c = 3, d = 4;

    id = shmget(KEY, sizeof(int), IPC_CREAT | PERMS); //Create an IPC of 4 bytes (=sizeof int) with shared memory (=allocate shared
    id2 = shmget(KEY+1, sizeof(int), IPC_CREAT | PERMS); //We change the key to avoid having a conflict with ptr
    ptr = (int *) shmat(id, NULL, 0); //Attach an existing shared memory (id) to an adress space (ptr)
    ptr2 = (int *) shmat(id2, NULL, 0);


    if (fork() == 0) //Creating a child process
    {//Parent
        (*ptr) = a + b;
        (*ptr2) = c + d;
        exit(0); //Close the parent process
    }
    else
    { //Child
        wait(NULL); //Wait until the parent process finish
        total = (*ptr) + (*ptr2);
        printf("Value of the addition 1 + 2 + 3 + 4 = %d\n", total); //Print the value of the total
        exit(0);
    }
}
```

Parallel Computing

Write a program that computes the folowing expression `(a + b) * (c - d) + (e + f)` using 3 different process.

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

int main(int argc, char **argv)
{
    int id, id2, id3;
    int *ptr, *ptr2, *ptr3;
    int total;
    int a = 1, b = 2, c = 3, d = 4, e = 5, f = 6;

    id = shmget(KEY, sizeof(int), IPC_CREAT | PERMS); //Create an IPC of 4 bytes (=sizeof int) with shared memory (=allocate shared
    id2 = shmget(KEY+1, sizeof(int), IPC_CREAT | PERMS); //We change the key to avoid having a conflict with ptr
    id3 = shmget(KEY+2, sizeof(int), IPC_CREAT | PERMS);



    ptr = (int *) shmat(id, NULL, 0); //Attach an existing shared memory (id) to an adress space (ptr)
    ptr2 = (int *) shmat(id2, NULL, 0);
    ptr3 = (int *) shmat(id2, NULL, 0);

    if (fork() == 0) //Creating a child process
    {//Parent
        if (fork() == 0)
        {
            (*ptr) = a + b;
            exit(0);
        }
        else
        {
            (*ptr2) = c - d;
            wait(NULL);
        }

        (*ptr3) = e + f;
        exit(0); //Close the parent process
    }
    else
    { //Child
        wait(NULL); //Wait until the parent process finish
        total = (*ptr) * (*ptr2) - (*ptr3);
        printf("Value of the equation (a + b) * (c - d) + (e + f) = %d\n", total); //Print the value of the total
        exit(0);
    }
}
```