LABORATORY 4
–
OPERATING SYSTEMS

RUFINO GARCIA SANCHEZ

Exercise 1

Write a child process which execute a process ./calc with argv[1], argv[2] and argv[3] parameters.

Where:

argv[1] - first number,

argv[2] - operation (+-x/),

argv[3] - second number.

Use one of the **execv* functions to pass arguments to the program.**

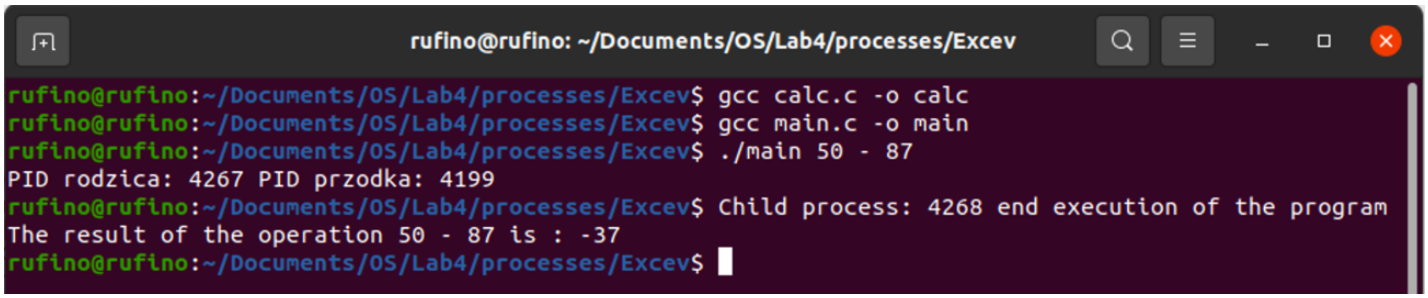
For this exercise this is the code I have made in the main.c to run with the execv* function:

```
include <stdio.h>
include <stdlib.h>
include <unistd.h>
include <wait.h>
include <string.h>


nt main(int argc, char* argv[])


if(argc !=4){
    printf("Not a suitable number of program parameters\n");
    return(1);
}

    // Write a child process which execute a process ./calc with
    //argv[1], argv[2] oraz argv[3] parameteres.
    //where argv[1] - first number,
    //argv[2] - operation  (+-x/) and
    // argv[3]  - second number
    //use table methods to pass argument to the program (argv)

    switch (fork()){
      //Child process
      case 0 :
        printf("Child process: %d end execution of the program\n", getpid());

        char*argv2[4];

        argv2[0] = "./calc";
        argv2[1] = argv[1];
        argv2[2] = argv[2];
        argv2[3] = argv[3];

        execv(argv2[0],argv2);

        exit(0);


      case -1 :
        printf("Error. Operation not possible due to fork function\n");
        exit(1);


      // Parent Process
      default:
        printf("PID rodzica: %d PID przodka: %d\n", getpid(), getppid());
    }
```

The code of calc.c is this:

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


int main(int argc, char* argv[]) {

 if(argc !=4){
   printf("Not a suitable number of program parameters\n");
   return(1);
 }

    int result =0;

    switch(argv[2][0]){
    case '+':
    result = atoi(argv[1]) + atoi(argv[3]);
    break;

    case '-':
    result = atoi(argv[1]) - atoi(argv[3]);
    break;

    case 'x':
    result = atoi(argv[1]) * atoi(argv[3]);
    break;

    case '/':
    if(atoi(argv[3])!=0){
        result = atoi(argv[1]) / atoi(argv[3]);
    }
    break;

    default:
     printf("Wrong operation selected\n");
     return(1);
    }

    printf("The result of the operation %s %s %s is : %d\n", argv[1], argv[2], argv[3], result);

    return 0;
}
```

And when we execute and run the code we can see in the next image that works correctly:



EXERCISE 2

Prepare 3 versions of this program using both execl* and execv* commands.

For the first one, **execlp*:**

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <wait.h>
#include <string.h>


int main(int argc, char* argv[])
{

 if(argc !=4){
    printf("Not a suitable number of program parameters\n");
    return(1);
 }

    switch (fork()){
      //Child process
      case 0 :
        printf("Child process: %d end execution of the program\n", getpid());
        char*argv2[4];

        argv2[0] = "./calc";
        argv2[1] = argv[1];
        argv2[2] = argv[2];
        argv2[3] = argv[3];

        execlp(argv2[0],argv2[0],argv2[1],argv2[2],argv2[3],(char*)NULL);

        exit(0);


      case -1 :
        printf("Error. Operation not possible due to fork function\n");
        exit(1);


      // Parent Process
      default:
        printf("PID rodzica: %d PID przodka: %d\n", getpid(), getppid());
    }

 return 0;
}
```

The code of the program **calc.c** still the same as the **EXERCISE 1.**

And when we execute and run the code we can see in the next image that works correctly:

```
rufino@rufino: ~/Documents/OS/Lab4/processes/Execlp
rufino@rufino:~/Documents/OS/Lab4/processes/Execlp$ gcc main.c -o mainclp
rufino@rufino:~/Documents/OS/Lab4/processes/Execlp$ gcc calc.c -o calc
rufino@rufino:~/Documents/OS/Lab4/processes/Execlp$ ./mainclp 23 + 76
PID rodzica: 10124 PID przodka: 9784
Child process: 10125 end execution of the program
The result of the operation 23 + 76 is : 99
```

For the second one, **execvp*:**

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <wait.h>
#include <string.h>


int main(int argc, char* argv[])
{

 if(argc !=4){
    printf("Not a suitable number of program parameters\n");
    return(1);
 }

    switch (fork()){
      //Child process
      case 0 :
        printf("Child process: %d end execution of the program\n", getpid());

        char*argv2[4];

        argv2[0] = "./calc";
        argv2[1] = argv[1];
        argv2[2] = argv[2];
        argv2[3] = argv[3];

        execvp(argv2[0],argv2);

        exit(0);


      case -1 :
        printf("Error. Operation not possible due to fork function\n");
        exit(1);


      // Parent Process
      default:
        printf("PID rodzica: %d PID przodka: %d\n", getpid(), getppid());
    }

 return 0;
}
```

The code of the program **calc.c** still the same as the **EXERCISE 1.**

And when we execute and run the code we can see in the next image that works correctly:

```
rufino@rufino: ~/Documents/OS/Lab4/processes/Execvp
rufino@rufino:~/Documents/OS/Lab4/processes/Execvp$ gcc main.c -o maincvp
rufino@rufino:~/Documents/OS/Lab4/processes/Execvp$ gcc calc.c -o calc
rufino@rufino:~/Documents/OS/Lab4/processes/Execvp$ ./maincvp 3 x 200
PID rodzica: 5473 PID przodka: 5006
Child process: 5474 end execution of the program
rufino@rufino:~/Documents/OS/Lab4/processes/Execvp$ The result of the operation 3 x 200 is : 600
```

For the third and last one, **execl*:**

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <wait.h>
#include <string.h>


int main(int argc, char* argv[])
{

 if(argc !=4){
    printf("Not a suitable number of program parameters\n");
    return(1);
 }

    switch (fork()){
      //Child process
      case 0 :
        printf("Child process: %d end execution of the program\n", getpid());
        char*argv2[4];

        argv2[0] = "./calc";
        argv2[1] = argv[1];
        argv2[2] = argv[2];
        argv2[3] = argv[3];

        execl(argv2[0],argv2[0],argv2[1],argv2[2],argv2[3],(char*)NULL);

        exit(0);


      case -1 :
        printf("Error. Operation not possible due to fork function\n");
        exit(1);


      // Parent Process
      default:
        printf("PID rodzica: %d PID przodka: %d\n", getpid(), getppid());
    }

 return 0;
}

```

The code of the program **calc.c** still the same as the **EXERCISE 1.**

And when we execute and run the code we can see in the next image that works correctly

```
rufino@rufino: ~/Documents/OS/Lab4/processes/Execl
rufino@rufino:~/Documents/OS/Lab4/processes/Execl$ gcc main.c -o maincl
rufino@rufino:~/Documents/OS/Lab4/processes/Execl$ gcc calc.c -o calc
rufino@rufino:~/Documents/OS/Lab4/processes/Execl$ ./maincl  89 / 1
PID rodzica: 9286 PID przodka: 6602
Child process: 9287 end execution of the program
rufino@rufino:~/Documents/OS/Lab4/processes/Execl$ The result of the operation 89 / 1 is : 89
```