

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.

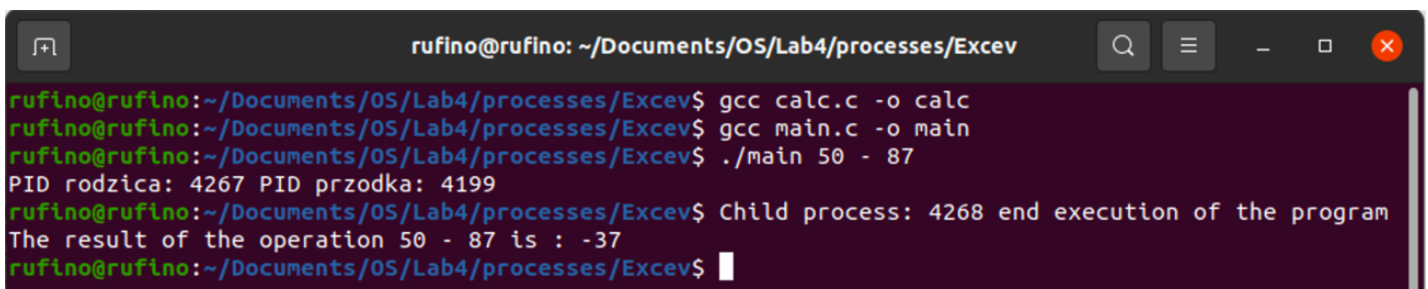
For this exercise this is the code I have made in the `main.c` to run with the `execv*` function:

```
1  include <stdio.h>
2  include <stdlib.h>
3  include <unistd.h>
4  include <wait.h>
5  include <string.h>
6
7
8  int main(int argc, char* argv[])
9  {
10
11     if(argc !=4){
12         printf("Not a suitable number of program parameters\n");
13         return(1);
14     }
15
16     // Write a child process which execute a process ./calc with
17     //argv[1], argv[2] oraz argv[3] paramteres.
18     //where argv[1] - first number,
19     //argv[2] - operation (+-x/) and
20     // argv[3] - second number
21     //use table methods to pass argument to the program (argv)
22
23     switch (fork()){
24         //Child process
25         case 0 :
26             printf("Child process: %d end execution of the program\n", getpid());
27
28             char*argv2[4];
29
30             argv2[0] = "./calc";
31             argv2[1] = argv[1];
32             argv2[2] = argv[2];
33             argv2[3] = argv[3];
34
35             execv(argv2[0],argv2);
36
37             exit(0);
38
39
40         case -1 :
41             printf("Error. Operation not possible due to fork function\n");
42             exit(1);
43
44         // Parent Process
45         default:
46             printf("PID rodzica: %d PID przodka: %d\n", getpid(), getppid());
47
48     }
```

The code of calc.c is this:

```
1  #include <stdio.h>
2  #include <sys/types.h>
3  #include <sys/stat.h>
4  #include <stdlib.h>
5  #include <unistd.h>
6
7
8  int main(int argc, char* argv[]) {
9
10     if(argc !=4){
11         printf("Not a suitable number of program parameters\n");
12         return(1);
13     }
14
15     int result =0;
16
17     switch(argv[2][0]){
18     case '+':
19         result = atoi(argv[1]) + atoi(argv[3]);
20         break;
21
22     case '-':
23         result = atoi(argv[1]) - atoi(argv[3]);
24         break;
25
26     case 'x':
27         result = atoi(argv[1]) * atoi(argv[3]);
28         break;
29
30     case '/':
31         if(atoi(argv[3])!=0){
32             result = atoi(argv[1]) / atoi(argv[3]);
33         }
34         break;
35
36     default:
37         printf("Wrong operation selected\n");
38         return(1);
39     }
40
41     printf("The result of the operation %s %s %s is : %d\n", argv[1], argv[2], argv[3], result);
42
43     return 0;
44 }
```

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



```
rufino@rufino: ~/Documents/OS/Lab4/processes/Excev
rufino@rufino:~/Documents/OS/Lab4/processes/Excev$ gcc calc.c -o calc
rufino@rufino:~/Documents/OS/Lab4/processes/Excev$ gcc main.c -o main
rufino@rufino:~/Documents/OS/Lab4/processes/Excev$ ./main 50 - 87
PID rodzica: 4267 PID przodka: 4199
rufino@rufino:~/Documents/OS/Lab4/processes/Excev$ Child process: 4268 end execution of the program
The result of the operation 50 - 87 is : -37
rufino@rufino:~/Documents/OS/Lab4/processes/Excev$
```

EXERCISE 2

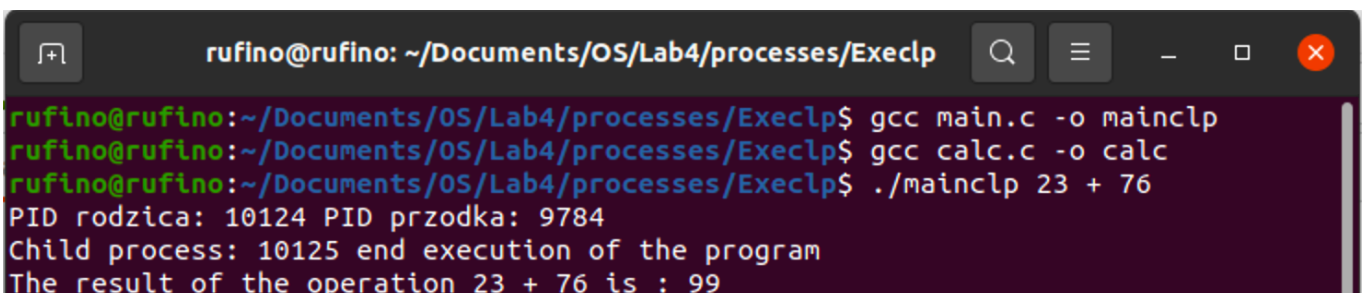
Prepare 3 versions of this program using both `execl*` and `execv*` commands.

For the first one, `execlp*`:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <wait.h>
5  #include <string.h>
6
7
8  int main(int argc, char* argv[])
9  {
10
11     if(argc !=4){
12         printf("Not a suitable number of program parameters\n");
13         return(1);
14     }
15
16     switch (fork()){
17         //Child process
18         case 0 :
19             printf("Child process: %d end execution of the program\n", getpid());
20             char*argv2[4];
21
22             argv2[0] = "./calc";
23             argv2[1] = argv[1];
24             argv2[2] = argv[2];
25             argv2[3] = argv[3];
26
27             execlp(argv2[0],argv2[0],argv2[1],argv2[2],argv2[3],(char*)NULL);
28
29             exit(0);
30
31
32         case -1 :
33             printf("Error. Operation not possible due to fork function\n");
34             exit(1);
35
36
37         // Parent Process
38         default:
39             printf("PID rodzica: %d PID przodka: %d\n", getpid(), getppid());
40     }
41
42     return 0;
43 }
```

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:



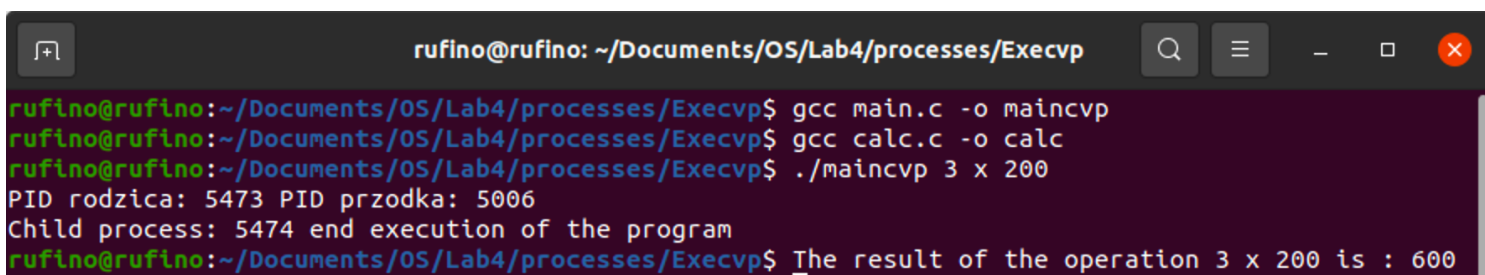
```
rufino@rufino: ~/Documents/OS/Lab4/processes/Execlp
rufino@rufino:~/Documents/OS/Lab4/processes/Execlp$ gcc main.c -o mainclp
rufino@rufino:~/Documents/OS/Lab4/processes/Execlp$ gcc calc.c -o calc
rufino@rufino:~/Documents/OS/Lab4/processes/Execlp$ ./mainclp 23 + 76
PID rodzica: 10124 PID przodka: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

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

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <wait.h>
5  #include <string.h>
6
7
8  int main(int argc, char* argv[])
9  {
10
11     if(argc !=4){
12         printf("Not a suitable number of program parameters\n");
13         return(1);
14     }
15
16     switch (fork()){
17         //Child process
18         case 0 :
19             printf("Child process: %d end execution of the program\n", getpid());
20
21             char*argv2[4];
22
23             argv2[0] = "./calc";
24             argv2[1] = argv[1];
25             argv2[2] = argv[2];
26             argv2[3] = argv[3];
27
28             execvp(argv2[0],argv2);
29
30             exit(0);
31
32
33         case -1 :
34             printf("Error. Operation not possible due to fork function\n");
35             exit(1);
36
37         // Parent Process
38         default:
39             printf("PID rodzica: %d PID przodka: %d\n", getpid(), getppid());
40
41     }
42
43     return 0;
44 }
```

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:



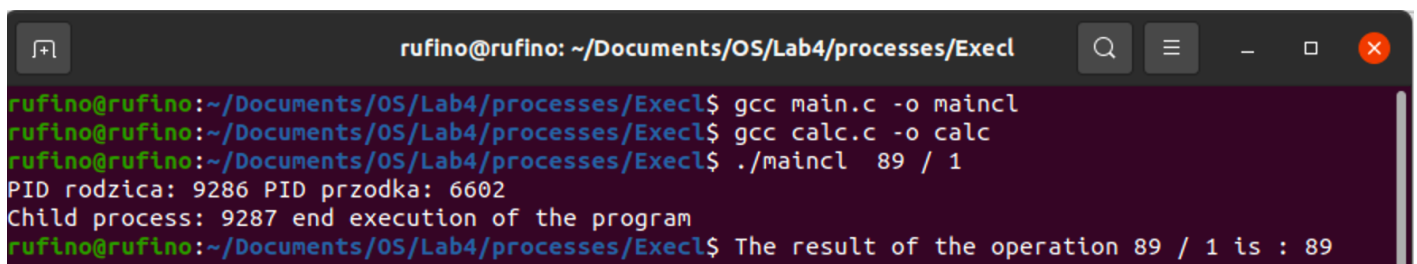
```
rufino@rufino: ~/Documents/OS/Lab4/processes/Execvp
rufino@rufino:~/Documents/OS/Lab4/processes/Execvp$ gcc main.c -o maincnp
rufino@rufino:~/Documents/OS/Lab4/processes/Execvp$ gcc calc.c -o calc
rufino@rufino:~/Documents/OS/Lab4/processes/Execvp$ ./maincnp 3 x 200
PID rodzica: 5473 PID przodka: 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***:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <wait.h>
5  #include <string.h>
6
7
8  int main(int argc, char* argv[])
9  {
10
11     if(argc !=4){
12         printf("Not a suitable number of program parameters\n");
13         return(1);
14     }
15
16     switch (fork()){
17         //Child process
18         case 0 :
19             printf("Child process: %d end execution of the program\n", getpid());
20             char*argv2[4];
21
22             argv2[0] = "./calc";
23             argv2[1] = argv[1];
24             argv2[2] = argv[2];
25             argv2[3] = argv[3];
26
27             execl(argv2[0],argv2[0],argv2[1],argv2[2],argv2[3],(char*)NULL);
28
29             exit(0);
30
31
32         case -1 :
33             printf("Error. Operation not possible due to fork function\n");
34             exit(1);
35
36
37         // Parent Process
38         default:
39             printf("PID rodzica: %d PID przodka: %d\n", getpid(), getppid());
40     }
41
42     return 0;
43 }
44
```

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
```