

Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №6
по дисциплине: **ОСИСП**
Тема: Средства межпроцессного взаимодействия

Выполнил
студент 2 курса
Корнаसेвич И. Д.

Проверил
Давидюк Ю. И.

Задание

- Программные каналы
- Родитель передает потомку три стороны треугольника, потомок возвращает его периметр

Написать программу, которая порождает дочерний процесс, и общается с ним через средства взаимодействия согласно варианту, передавая и получая информацию согласно варианту. Передачу и получение информации каждым из процессов сопровождать выводом на экран информации типа "процесс такой-то передал/получил такую-то информацию". Сообщение вводит пользователь через терминал. Дочерние процессы начинают операции после получения сигнала SIGUSR1 от родительского процесса. После отработки дочерний процесс должен возвращать результат родительскому процессу!

main.c

```
1  #include<stdio.h>
2  #include<signal.h>
3  #include<unistd.h>
4  #include <stdlib.h>
5
6  void sig_handler_parent(int signum) {
7      printf("Parent : Received a response signal from child \n");
8  }
9
10 void sig_handler_child(int signum) {
11     printf("Child : Received a signal from parent \n");
12 }
13
14 int main() {
15     int fd[2];
16
17     if(pipe(fd) < 0){
18         printf("Can\'t create pipe\n");
19         return -1;
20     }
21
22     pid_t pid;
23     if ((pid = fork()) < 0) {
24         printf("Fork Failed\n");
25         exit(1);
26     }
27     /* Child Process */
28     else if (pid == 0) {
29         signal(SIGUSR1, sig_handler_child);
30         pause();
31         double sides[3];
32         read(fd[0], sides, 3*sizeof(double));
33         close(fd[0]);
34         sides[0] += sides[1] + sides[2];
35         write(fd[1], sides, sizeof(double));
36         close(fd[1]);
37         printf("Child: sending signal to Parent\n");
```

```

38     kill(getppid(), SIGUSR1);
39 }
40     /* Parent Process */
41 else {
42     signal(SIGUSR1, sig_handler_parent);
43     double sides[3] = {100, 200, 150};
44     write(fd[1], sides, 3*sizeof(double));
45     close(fd[1]);
46     printf("Parent: sending signal to Child\n");
47     kill(pid, SIGUSR1);
48     printf("Parent: waiting for response\n");
49     pause();
50     double side[1];
51     read(fd[0], side, sizeof(double));
52     close(fd[0]);
53     printf("Parent: perimeter is: %f", side[0]);
54 }
55 return 0;
56 }

```

results.

```

1 Parent: sending signal to Child
2 Parent: waiting for response
3 Child : Received a signal from parent
4 Child: sending signal to Parent
5 Parent : Received a response signal from child
6 Parent: perimeter is: 450.000000
7 Process finished with exit code 0

```

Оказалось очень интересно, что получение сигнала прерывает pause().

Вывод: Я использовал pipe для передачи информации между родственными процессами, также давал команду процессам для начала работы используя сигналы и создал собственный обработчик сигналов.