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

Лабораторная работа №6
за 4 семестр
По дисциплине: «ОСиСП»
Тема: «СРЕДСТВА МЕЖПРОЦЕССНОГО ВЗАИМОДЕЙСТВИЯ»

Выполнил:
студент 2-ого курса
Шевчук А. В.
группы ПО-3
Проверила:
Давидюк Ю. И.

Брест-2020

Вариант 27

Задание для выполнения

Ознакомиться с руководством, теоретическими сведениями и лекционным материалом по использованию и функционированию средств взаимодействия. 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!

Средство взаимодействия	Действия
Именованные каналы	Родитель передает потомку три строки S1, S2 и S3, потомок возвращает их конкатенацию S2+S1+S3

Ход работы

Код программы:

```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
#include <signal.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <string.h>
#include <stdlib.h>
void my_handler(){
        printf("'Дочерний процесс может приступить к работе'\n");
}
int Exitt(char* msg){
        printf("%s\n", msg);
        return -1;
}
int main(){
        int i, fd, result, ppid, size, size2, size3, N;
        char name[]="NameFile.fifo";
        char simbol, string1[50], string2[50], string3[50], konkat[255], resultkonkat[255];
        (void)signal(SIGUSR1, my_handler);
        (void)umask(0);
        if ((size = read(0, string1, 255))>=50) Exitt("Строка больше допустимого размера!");
        if ((size2 = read(0, string2, 255))>=50) Exitt("Строка больше допустимого размера!");
        if ((size3 = read(0, string3, 255))>=50) Exitt("Строка больше допустимого размера!");
        if (mkfifo(name, 0666) < 0) Exitt("Can\'t create FIFO");
        if ((result = fork()) < 0) Exitt("Can\'t fork child!");
        else if (result > 0) {
                if((fd = open(name, O_WRONLY)) < 0) Exitt("Can\'t open FIFO for writing!");
                kill(result, SIGSTOP);
                printf("Дочерний процесс остановлен, т. к. в FIFO ещё не записана информация.\n");
                if ((write(fd, string1, size))<size) Exitt("Can\'t write all string to FIFO!");
                if ((write(fd, string2, size2))<size2) Exitt("Can\'t write all string to FIFO!");
                if ((write(fd, string3, size3))<size3) Exitt("Can\'t write all string to FIFO!");
                close(fd);
```

```
                printf("Родительский процесс записал информацию в FIFO.\n");
                kill(result, SIGUSR1);
                kill(result, SIGCONT);
                if ((fd = open(name, O_RDONLY))<0) Exitt("Can\'t open FIFO for reading!");
                N=0;
                while (read(fd, &resultkonkat[N++], 1));
                printf("Родительский процесс получил информацию из FIFO:\n");
                for (i=0; i<N-1; i++) printf("%c", resultkonkat[i]);
                printf("\n");
                close(fd);
        } else {
                ppid = getppid();
                if((fd = open(name, O_RDONLY))<0) Exitt("Can\'t open FIFO for reading!");
                N=0;
                printf("Родительский процесс остановлен.\n");
                kill(ppid, SIGSTOP);
                printf("\nДочерний процесс получил три предложения из FIFO и объедилил их в одно:\n");
                while (read(fd, &simbol, 1))
                {
                        if (simbol!='\n'){
                                konkat[N++]=simbol;
                                printf("%c", konkat[N-1]);
                        }
                }
                printf("\n");
                if ((fd = open(name, O_WRONLY))<0) Exitt("Can\t open FIFO for writing!");
                if ((write(fd, konkat, N))<N) Exitt("Can\'t write all string to FIFO!");
                printf("Дочерний процесс записал новую информацию в FIFO.\n");
                kill(ppid, SIGCONT);
                printf("Родительский процесс снова запущен.\n");
                close(fd);
        }
        unlink(name);
        return 0;
}
```

Результаты тестирования:

1)

```
andrey@andrey-VirtualBox:~$ gcc program.c -o program
andrey@andrey-VirtualBox:~$ ./program
qwerty
123456789
abc
Дочерний процесс остановлен, т. к. в FIFO ещё не записана информация.
Родительский процесс записал информацию в FIFO.
'Дочерний процесс может приступить к работе'
Родительский процесс остановлен.

Дочерний процесс получил три предложения из FIFO и объедилил их в одно:
qwerty123456789abc
Дочерний процесс записал новую информацию в FIFO.
Родительский процесс снова запущен.
Родительский процесс получил информацию из FIFO:
qwerty123456789abc
```

2)

```
andrey@andrey-VirtualBox:~$ ./program
1
12345678901234567890123456789012345678901234567890
Строка больше допустимого размера!
andrey@andrey-VirtualBox:~$
```

Вывод: написал программу, в которой родительский и дочерний процессы общаются между собой, используя как средство связи именованные каналы.