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

ОТЧЕТ
По лабораторной работе №6
Дисциплина «ОСиСП» за IV семестр

Выполнила:

студент группы ПО-3

Ковалёва А. И.

Проверила:

Давидюк Ю. И.

Брест 2020

Лабораторная работа 6
Средства межпроцессного взаимодействия

Цель:

Изучить работу с средствами межпроцессного взаимодействия в ОС Linux.

Вариант 12

Задание:

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!

12	Именованные каналы	Родитель передает потомку пять строк, потомок возвращает строки, которые содержат по одной цифре

Код программы:

```
#include <errno.h>
#include <sys/stat.h>
#include <string.h>
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdbool.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.h>

bool isStringWithOneNumber(char str[]) {
    char numbers[] = "123456789";
    int num = 0;
    for(int i = 0; i < strlen(str); i++) {
        for(int j = 0; j < strlen(numbers); j++) {
            if (str[i] == numbers[j]) {
                    num++;
            }
        }
    }
    return num == 1 ? true : false;
}
```

```
char resStrings[5][50];

void handler() {
    printf("Child pid: %d \n", getpid());
    int fd;             //for opening FIFO
    size_t size = 6;        //for write/read
    char name1[]="first.fifo";    //name 1 FIRO
    char name2[]="second.fifo";    //name 2 FIFO
    if((fd = open(name1, O_RDONLY)) < 0) {        //open 1 FIFO
        printf("Can not open FIFO for reading\n");
        exit(-1);
    }
    for (int i = 0; i < 5; i++) {
        read(fd, resStrings[i], size);
        printf("%s\n", resStrings[i]);
    }
    printf("\nChild read 5 strings\n");
    if (mknod(name2, S_IFIFO | 0666, 0) < 0) {    //create 2 FIFO
        printf("Can not create FIFO\n");
        exit(-1);
    }
    close(fd);
}

int main() {
    int fd;             //for opening FIFO
    size_t size = 6;        //for write/read
    char name1[]="first.fifo";    //name 1 FIRO
    char name2[]="second.fifo";    //name 2 FIFO
    (void)umask(0);
    system("rm *.fifo");
    if (mknod(name1, S_IFIFO | 0666, 0) < 0) {
        printf("Can not create FIFO\n");
        exit(-1);
    }
    int parent = fork();
    if (parent > 0) {
        sleep(1);
        kill(parent, SIGUSR1);
        sleep(2);
        if ((fd = open(name1, O_WRONLY)) < 0) {        //open 1 FIFO
            printf("Can not open FIFO for writing\n");
            exit(-1);
        }
        char strings[5][50];
        for (int i = 0; i < 5; i++) {
            printf("Eneter str(5 symbols): ");
            scanf("%s", strings[i]);
            strings[i][5] = '\0';
        }
        printf("\nParent (pid: %d) send 5 strings", getpid());
        for (int i = 0; i < 5; i++) {
            write(fd, strings[i], size);
        }
        close(fd);
        sleep(1);
        if((fd = open(name2, O_RDONLY)) < 0) {
            printf("Can not open FIFO for reading\n");
            exit(-1);
        }
```

```
        kill(parent, SIGCONT);
        sleep(4);
        //printf("Parent pid: %d \n", getpid());
        printf("Answer: \n");
        for(int i = 0; i < 5; i++) {
            char str[size];
            read(fd, str, size);
            printf("%s\n",str);
        }
        close(fd);
        printf("\nParent (pid: %d) exit\n", getpid());
    } else if (parent == 0) {
        signal(SIGUSR1, handler);
        pause();
        if ((fd = open(name2, O_WRONLY)) < 0) {        //open 2 FIFO
            printf("Can not open FIFO for writing\n");
            exit(-1);
        }
        for (int i = 0; i < 5; i++) {
            if (isStringWithOneNumber(resStrings[i])) {
                write(fd, resStrings[i], strlen(resStrings[i]));
                printf("%s write in second fifo\n", resStrings[i]);
            }
        }
        printf("%d", getpid());
        printf("%s\n", " give information result to parent");
        close(fd);
        exit(0);
    } else {
    printf("Can not fork parent\n");
    exit(-1);
    }
    return 0;
}
```

Результат выполнения:

```
Child pid: 52337
Eneter str(5 symbols): 12345
Eneter str(5 symbols): dsd2d
Eneter str(5 symbols): aaaaa
Eneter str(5 symbols): 7hhhh
Eneter str(5 symbols): wmdj6

Parent (pid: 52334) send 5 strings12345
dsd2d
aaaaa
7hhhh
wmdj6

Child read 5 strings
dsd2d write in second fifo
7hhhh write in second fifo
wmdj6 write in second fifo
52337 give information result to parent
Answer:
dsd2d7
hhhh
wmdj6
```

```
Parent (pid: 52334) exit
Program ended with exit code: 0
```

Вывод: изучила работу с средствами межпроцессного взаимодействия в ОС Linux.