**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра вычислительной техники**

отчет

**по лабораторной работе №7**

**по дисциплине «Организация процессов и программирование в среде Linux»**

Темы: ОБМЕН ДАННЫМИ ЧЕРЕЗ КАНАЛ

|  |  |  |
| --- | --- | --- |
| Студент гр. 8306 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Пеунов В.В. |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Разумовский Г.В. |

Санкт-Петербург,

2021

**Цель работы**

Знакомство с механизмом обмена данными через программный канал и системными вызовами, обеспечивающими такой обмен

**Задание на лабораторную работу**

1. Написать программу, которая обменивается данными через канал с двумя потомками. Программа открывает входной файл, построчно читает из него данные и записывает их в канал. Потомки выполняют свои программы и поочередно читают символы из канала и записывают их в свои выходные файлы: первый потомок – нечетные символы, а второй – четные. Синхронизация работы потомков должна осуществляться напрямую с использованием сигналов SIGUSR1 и SIGUSR2. Об окончании записи файла в канал программа оповещает потомков сигналом SIGQUIT и ожидает завершения работы потомков. Когда они заканчивают работу, программа закрывает канал.

2. Откомпилировать все программы и запустить их.

**Описание работы**

Содержание input:

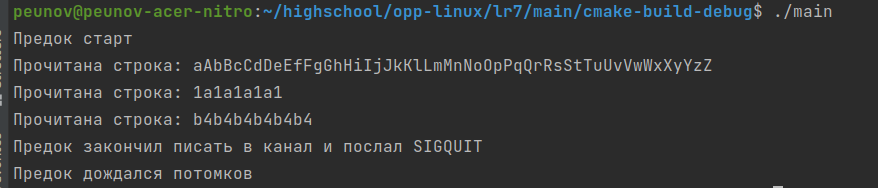
* aAbBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTuUvVwWxXyYzZ
* 1a1a1a1a1
* b4b4b4b4b4b4

Содержание outout1:

* abcdefghijklmnopqrstuvwxyz11111444444

Содержание output2:

* ABCDEFGHIJKLMNOPQRSTUVWXYZaaaabbbbbb

Рисунок 1. Вывод программы

**Вывод:** в ходе лабораторной работы осуществлено знакомство с механизмом обмена данными через программный канал и системными вызовами, обеспечивающими такой обмен

**Приложение A. Основная программа**

#include <iostream>

#include <sys/wait.h>

#include <unistd.h>

#include <cstring>

#include <fcntl.h>

#include <fstream>

#include <csignal>

using namespace std;

/\*

\* Написать программу, которая обменивается данными через канал с двумя потомками.

\* Программа открывает входной файл, построчно читает из него данные и записывает их в канал.

\* Потомки выполняют свои программы и поочередно читают символы из канала и записывают их в свои выходные файлы:

\* первый потомок – нечетные символы, а второй – четные.

\* Синхронизация работы потомков должна осуществляться напрямую с использованием сигналов SIGUSR1 и SIGUSR2.

\* Об окончании записи файла в канал программа оповещает потомков сигналом SIGQUIT и ожидает

\* завершения работы потомков.

\* Когда они заканчивают работу, программа закрывает канал.

\* \*/

#define SIZE 128

string inputFilePath = "/home/peunov/highschool/opp-linux/lr7/files/input";

const char \* childProgramPath = "/home/peunov/highschool/opp-linux/lr7/child/cmake-build-debug/child";

const char \* outputFirstFilePath = "/home/peunov/highschool/opp-linux/lr7/files/output1.txt";

const char \* outputSecondFilePath = "/home/peunov/highschool/opp-linux/lr7/files/output2.txt";

void signal\_handler(int sig){

if (sig == SIGQUIT) {

cout << "Родитель получил SIGQUIT" << endl;

}

if (sig == SIGUSR1) {

cout << "Родитель получил SIGUSR1" << endl;

}

if (sig == SIGUSR2) {

cout << "Родитель получил SIGUSR2" << endl;

}

}

void setSignals(){

signal(SIGQUIT, signal\_handler);

signal(SIGUSR1, signal\_handler);

signal(SIGUSR2, signal\_handler);

}

int main(int argc, char\*\* argv){

cout << "Предок старт" << endl;

int pipeFd[2];

string inputStr;

bool flag = true;

pipe(pipeFd);

pid\_t pid\_1 = fork();

if (!pid\_1) {

execl(childProgramPath,"1", &pipeFd[0], outputFirstFilePath, NULL);

}

pid\_t pid\_2 = fork();

if (not pid\_2) {

execl(childProgramPath, "2", &pipeFd[0], outputSecondFilePath, NULL);

}

sleep(1);

setSignals();

ifstream input;

input.open(inputFilePath, ios\_base::in);

fcntl(\*pipeFd, F\_SETFL, O\_NONBLOCK);

kill(pid\_1, SIGUSR2);

while (getline(input, inputStr)) {

cout << "Прочитана строка: " << inputStr << endl;

write(pipeFd[1], inputStr.c\_str(), inputStr.length());

}

cout << "Предок закончил писать в канал и послал SIGQUIT" << endl;

kill(pid\_1, SIGINT);

kill(pid\_2, SIGINT);

waitpid(pid\_1, nullptr, 0);

waitpid(pid\_2, nullptr, 0);

cout << "Предок дождался потомков" << endl;

close(pipeFd[0]);

close(pipeFd[1]);

input.close();

return 0;

}

**Приложение B. Заменяемая программа**

include <cstdio>

#include <cstdlib>

#include <unistd.h>

#include <csignal>

#include <iostream>

#include <fstream>

using namespace std;

ofstream output;

int readPipeFd, childNumber;

bool readFlag = true;

bool doneWhite = false;

void end\_write\_signal(int sig){

cout << "Запись в канал закончена" << endl;

doneWhite = true;

}

void read\_for\_pipe(int signal){

cout << "Поток " << childNumber << " получил сигнал" << endl;

char symbol;

if(read(readPipeFd, &symbol, 1) != -1){

cout << "Потомок " << childNumber << ". Прочитанный символ: " << symbol << endl;

output << symbol;

} else {

cout << "Потомок " << childNumber << ": канал пуст" << endl;

if(doneWhite) readFlag = false;

}

kill(0, signal);

}

void first\_signal(int sig){

//если потомок получает свой же сигнал просто выходим

if(childNumber == 1){

//cout << "Потомок 1 получил свой же SIGUSR1" << endl;

return;

}

// иначе читаем из канала

read\_for\_pipe(SIGUSR2);

}

void second\_signal(int sig){

//если потомок получает свой же сигнал просто выходим

if(childNumber == 2){

//cout << "Потомок 2 получил свой же SIGUSR2" << endl;

return;

}

// иначе читаем из канала

read\_for\_pipe(SIGUSR1);

}

void setSignals(){

signal(SIGINT, end\_write\_signal);

signal(SIGUSR1, first\_signal);

signal(SIGUSR2, second\_signal);

}

int main(int argc, char\*\* argv){

if(argc != 3){

printf("Недостаточно аргументов для запуска\n");

return 0;

}

childNumber = atoi(argv[0]);

readPipeFd = \*argv[1];

char\* fileOutputPath = argv[2];

cout << "Запущен потомок : " << childNumber << endl;

setSignals();

output.open(fileOutputPath, ios::out);

while(readFlag)

pause();

output.close();

cout << "Выход из потомка " << childNumber << endl;

exit(EXIT\_SUCCESS);

}