## 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»

**Тема: Обмен данными через канал**

|  |  |
| --- | --- |
| Студент гр. 8308 | Петров Г.А. |
| Преподаватель | Разумовский Г.В. |

Санкт-Петербург 2021

# Цель работы

Целью лабораторной работы является знакомство с механизмом обмена данными через программный канал и системными вызовами, обеспечивающими такой обмен.

# Задание

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.

# Примеры выполнения программы

Программа была разработана и откомпилирована. После чего программа была запушена. Результат работы программы приведен на рисунке 1. Также представлены распечатки файлов TEXT.txt, CHILD1.txt и CHILD2.txt.

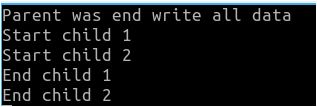


Рисунок 1

TEXT.txt

C:\Users\Admin\Desktop\Linuxshare\lab7\2.JPG

CHILD1.txt

C:\Users\Admin\Desktop\Linuxshare\lab7\3.JPG

CHILD2.txt

C:\Users\Admin\Desktop\Linuxshare\lab7\4.JPG

# Исходный код программ

## Lab7.cpp

#include <iostream>

#include <fstream>

#include <signal.h>

#include <string.h>

#include <unistd.h>

#include <wait.h>

#include <fcntl.h>

const char\* ofname1="CHILD1.txt";

const char\* ofname2="CHILD2.txt";

FILE\* ifile = fopen("TEXT.txt","r");

//передача данных по каналу

int main()

{

int fildes[2];

pid\_t pid1,pid2;

char buf[64];

//добавление в маску проца необходимых сигналов синхронизации

sigset\_t mask;

sigaddset(&mask,SIGQUIT);

sigaddset(&mask,SIGUSR1);

sigaddset(&mask,SIGUSR2);

sigprocmask(SIG\_BLOCK,&mask,NULL);

//создание канала

if(pipe2(fildes,O\_NONBLOCK)==-1)

{// массив fildes, [0] для чтения и [1] для записи

exit(001);

}

else

{

if(!(pid1=fork()))

{//создание потомка 1

std::cout<<"Start child 1"<<std::endl;

close(fildes[1]);

execl("child1","child1",&fildes[0],ofname1,NULL);

}

else

{//создание потомка 2

if(!(pid2=fork()))

{

std::cout<<"Start child 2"<<std::endl;

close(fildes[1]);

execl("child2","child2",&fildes[0],ofname2,NULL);

}

}

//запись данных из файла в канал

while(fgets(buf,64,ifile))

{

write(fildes[1],buf,strlen(buf));

}

//после записи данных передача сигнала об окончании записи

std::cout<<"Parent was end write all data"<<std::endl;

kill(0,SIGQUIT);

//ожидание завершения потомков

waitpid(pid1,NULL,0);

std::cout<<"End child 1"<<std::endl;

waitpid(pid2,NULL,0);

std::cout<<"End child 2"<<std::endl;

//закрытие канала

sigprocmask(SIG\_UNBLOCK,&mask,NULL);

close(fildes[0]);

close(fildes[1]);

}

return 0;

}

## Child1.cpp

#include <iostream>

#include <fstream>

#include <unistd.h>

#include <signal.h>

#include <string.h>

//потомок общения через канал

int main(int argc, char\* argv[])

{

//получение имени файла записи и канала для чтения

std::ofstream fout(argv[2]);

int read\_pipe = \*argv[1];

char ch;

int sig;

//добавление сигнала в маску проца

sigset\_t mask;

sigaddset(&mask,SIGUSR1);

//считывать посимвольно из канала пока есть что

while(read(read\_pipe,&ch,1)>0)

{

fout<<ch;//запись в файл

//std::cout<<"Р1: "<<ch<<std::endl;

kill(0,SIGUSR2);//передача разрешения 2 потомку

sigwait(&mask,&sig);//ожидание разрешения от 2 потомка

}

kill(0,SIGUSR2);

fout.close();

exit(0);

}

## Child2.cpp

#include <iostream>

#include <fstream>

#include <unistd.h>

#include <signal.h>

#include <string.h>

//потомок 2 чтения из канала

int main(int argc, char\* argv[])

{

//имя файла и канал чтения

std::ofstream fout(argv[2]);

int read\_pipe = \*argv[1];

char ch;

int sig;

//добавление сиганла потомку

sigset\_t mask;

sigaddset(&mask,SIGUSR2);

//потомок 2 читает ждет потомка 1

sigwait(&mask,&sig);

while(read(read\_pipe,&ch,1)>0)

{//читать из канала пока есть что

fout<<ch;//запись в файл

//std::cout<<"Р2: "<<ch<<std::endl;

kill(0,SIGUSR1);//передача разрешения потомку 1

sigwait(&mask,&sig);//ожидания разрешения от потомка 1

}

fout.close();

exit(0);

}

# Вывод

При выполнении лабораторной работы изучены и использованы механизм обмена данными через программный канал и системные вызовы, обеспечивающие такой обмен. Программа разработанная в соответствии с заданием, работает корректно.