|  |
| --- |
| «Санкт-Петербургский государственный электротехнический университет |
| «ЛЭТИ» им. В.И.Ульянова(Ленина)» |
| (СПбГЭТУ «ЛЭТИ») Кафедра вычислительной техники |
|  |
| **Отчет**  **по лабораторным работам № 7**  **по дисциплине** **«Организация процессов и программирование в среде Linux»** |
| Тема: «Обмен данными через канал» |
|  |

|  |
| --- |
|  |

|  |  |
| --- | --- |
| Студент гр. 9307 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_Брызгалова Е. А. |
|  |  |
|  |  |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Разумовский Г. В. |

Оглавление

[Цель работы: 2](#_Toc118034666)

[Задание 2](#_Toc118034667)

[Распечатки входных и выходного файлов. 2](#_Toc118034668)

[Текст программы main.cpp 3](#_Toc118034669)

[Вывод 6](#_Toc118034670)

Цель работы:

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

Задание

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

Распечатки входных и выходного файлов.

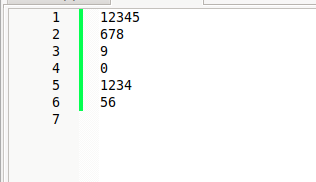


Рисунок 1. Файл input

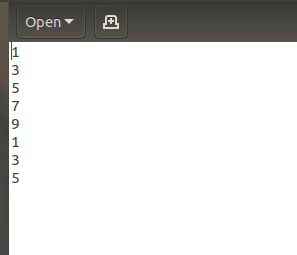


Рисунок 2. Файл 1

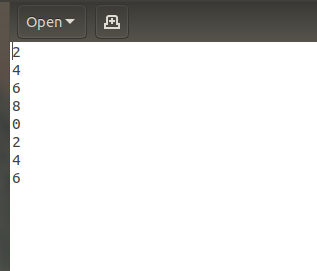


Рисунок 3. Файл 2

# **Текст программы main.cpp**

#include <iostream>

#include <fstream>

#include <unistd.h>

#include <cstring>

#include <sys/wait.h>

#include <signal.h>

#include <wait.h>

using namespace std;

int main() {

sigset\_t mask;

sigemptyset(&mask);

sigaddset(&mask, SIGUSR1);

sigaddset(&mask, SIGUSR2);

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

int pipefd[2];

pid\_t pid1,pid2;

if (pipe(pipefd) == -1 ) {

exit(EXIT\_FAILURE);

}

pid1 = fork();

if (pid1 == 0) {

close(pipefd[1]);

execl("c1","c1", "output1.txt", &pipefd[0], NULL);

}

else {

pid2 = fork();

if (pid2 == 0) {

close(pipefd[1]);

execl("c2","c2", "output2.txt", &pipefd[0], NULL);

}

else {

close(pipefd[0]);

char ch;

cout << "Parent: Wait children ready" << endl;

sleep(1);

ifstream file1("text.txt");

if (file1.is\_open()) {

string s;

while(getline(file1, s)){

write(pipefd[1], &s[0], strlen(s.c\_str()));

cout << "Parent: " << s << endl;

}

cout << "Send to children SIGQUIT" << endl;

kill(pid1, SIGQUIT);

kill(pid2, SIGQUIT);

file1.close();

}

close(pipefd[1]);

int st1, st2;

waitpid(pid1, &st1, 0);

waitpid(pid2, &st2, 0);

exit(EXIT\_SUCCESS);

}

}

return 0;

}

# **Текст программы с1.cpp**

#include <iostream>

#include <fstream>

#include <unistd.h>

#include <cstring>

#include <sys/wait.h>

#include <signal.h>

#include <wait.h>

#include <fcntl.h>

using namespace std;

int fd;

int flags;

bool obtain\_sigquit = false;

void sigquit\_handler(int signal) {

obtain\_sigquit = true;

cout << "Child1 SIGQUIT " << endl;

}

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

signal(SIGQUIT, sigquit\_handler);

sigset\_t mask;

sigemptyset(&mask);

sigaddset(&mask, SIGUSR1);

cout << "Child1 started" << endl;

fd = \*argv[2];

lockf(fd, F\_LOCK, 0);

int sig;

ofstream outstream;

outstream.open(argv[1], ios::app);

char c;

while(true) {

int l = read(fd, &c, 1);

if (l <= 0 && obtain\_sigquit) {

break;

}

outstream << c << endl;

cout << "Child1: " << c << endl;

killpg(0, SIGUSR2);

sigwait(&mask, &sig);

}

killpg(0, SIGUSR2);

return 0;

}

# **Текст программы с2.cpp**

#include <iostream>

#include <fstream>

#include <unistd.h>

#include <cstring>

#include <sys/wait.h>

#include <signal.h>

#include <wait.h>

#include <fcntl.h>

using namespace std;

int fd;

int flags;

bool obtain\_sigquit = false;

void sigquit\_handler(int signal) {

obtain\_sigquit = true;

cout << "Child12 SIGQUIT " << endl;

}

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

signal(SIGQUIT, sigquit\_handler);

sigset\_t mask;

sigemptyset(&mask);

sigaddset(&mask, SIGUSR1);

cout << "Child2 started" << endl;

fd = \*argv[2];

lockf(fd, F\_LOCK, 0);

int sig;

ofstream outstream;

outstream.open(argv[1], ios::app);

char c;

while(true) {

int l = read(fd, &c, 1);

if (l <= 0 && obtain\_sigquit) {

break;

}

outstream << c << endl;

cout << "Child2: " << c << endl;

killpg(0, SIGUSR2);

sigwait(&mask, &sig);

}

killpg(0, SIGUSR1);

return 0;

}

Вывод

В процессе выполнения лабораторной работы познакомились с механизмом обмена данными через программный канал и системными вызовами, обеспечивающими такой обмен.