МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

## Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и технологий

**Отчет по лабораторной работе № 10**

по дисциплине:” Системное программирование”

на тему: “РАБОТА С КАНАЛАМИ ”

Выполнил**:** студент группы 10701321

Мацкевич М.Ю.

Принял**:** Кондратёнок Е.В.

Минск 2023

# Лабораторная работа № 8. РАБОТА С КАНАЛАМИ

# Цель работы: Освоить механизм взаимодействия между процессами на основе каналов (pipes)

**Задание 1**

Передайте через неименованный и именованный каналы следующую информацию : неделя семестра получить имя операционной системы, версии и реализацию ядра, системную дату и время, а также календарь

**Решение**

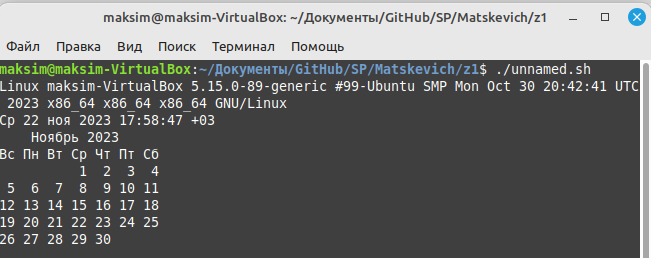
Скрипт для передачи через не именованный канал  
{ uname -a; date; cal; } | cat

Скрипт для передачи информации в именованный канал  
mkfifo pipe

echo -e "$(uname -a) \n$(date) \n$(cal)" > pipe

Скрипт для получения из именованного канала   
cat pipe

rm pipe

Рис.1 Передача информации через не именованный канал

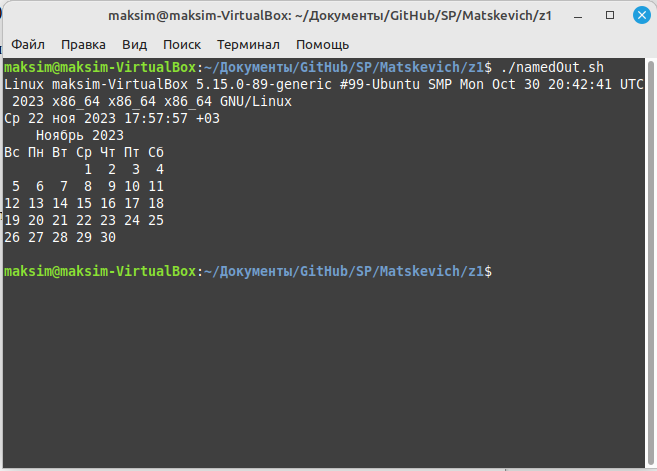


Рис.2 – Передача информации через именованный канал

**Задание 1**

Реализуйте взаимодействия двух процессов-братьев через канал, созданный в родительском процессе, с использованием системных вызовов .

**Решение**

Напишем код для родителя  
**parent.cpp**

#include <iostream>

#include <unistd.h>

#include <sys/wait.h>

#include <cstring>

#define STR\_SIZE 32

int main() {

int pf[2];

int pid1, pid2;

char spf[2][STR\_SIZE];

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

std::cerr << "pipe() error\n";

return 1;

}

snprintf(spf[0], sizeof(spf[0]), "%d", pf[0]);

snprintf(spf[1], sizeof(spf[1]), "%d", pf[1]);

if ((pid1 = fork()) == 0) {

close(pf[0]);

execl("./pipeInput", "pipeInput", spf[1], nullptr);

std::cerr << "exec() [src] error\n";

return 1;

}

if ((pid2 = fork()) == 0) {

close(pf[1]);

execl("./pipeOutput", "pipeOutput", spf[0], nullptr);

std::cerr << "exec() [dst] error\n";

return 1;

}

waitpid(pid1, nullptr, 0);

close(pf[0]);

close(pf[1]);

waitpid(pid2, nullptr, 0);

return 0;

}

Одного дочернего класса

**child1.cpp**  
#include <iostream>

#include <string>

#include <cstdlib>

#include <cstring>

#include <unistd.h>

#include <sys/utsname.h>

#include <ctime>

using namespace std; // Глобальная директива using

#define WAIT\_SECS 2

string executeCommand(const char\* cmd);

string getInfo();

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

int i, fd;

if (argc < 2) {

cerr << "src: Too few arguments\n";

return 1;

}

fd = atoi(argv[1]);

cerr << "Wait please";

for (i = 0; i < WAIT\_SECS; i++, sleep(1))

cerr << ".";

cerr << "\n";

string info = getInfo();

if (write(fd, info.c\_str(), info.length()) == -1) {

cerr << "src: write() error\n";

return 1;

}

close(fd);

return 0;

}

string executeCommand(const char\* cmd) {

string result = "";

FILE\* pipe = popen(cmd, "r");

if (!pipe) {

cerr << "popen() failed\n";

return result;

}

char buffer[128];

while (fgets(buffer, sizeof(buffer), pipe) != nullptr) {

result += buffer;

}

pclose(pipe);

return result;

}

string getInfo() {

time\_t now = time(0);

tm\* localTime = localtime(&now);

char buffer[80];

strftime(buffer, sizeof(buffer), "%Y-%m-%d", localTime);

string date = buffer;

strftime(buffer, sizeof(buffer), "%H:%M:%S", localTime);

string time = buffer;

utsname unameData;

uname(&unameData);

string osname = unameData.sysname;

string osVersion = unameData.release;

string calendar = executeCommand("cal");

return "OS Name: " + osname + "OS Version: " + osVersion +

"Current Date and Time: " + date + " " + time + "\n" +

"Calendar:\n" + calendar;

}

И второго  
**child2.cpp**

#include <iostream>

#include <cstdlib>

#include <unistd.h>

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

int fd;

char ch;

if (argc < 2) {

std::cerr << "dst: Too few arguments\n";

return 1;

}

fd = atoi(argv[1]);

while (read(fd, &ch, 1) > 0)

write(1, &ch, 1);

close(fd);

return 0;

}

В родителе у нас запустится 2 процесса, первый передаст информацию в канал, второй получит из него.

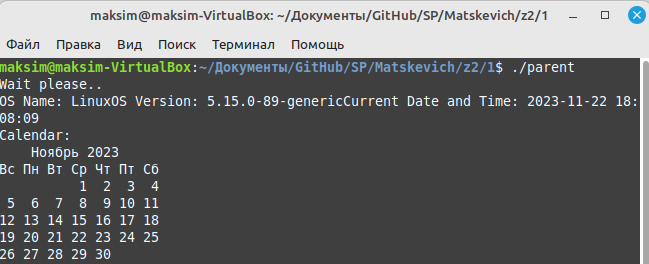
В

Рис.3 Передача информации между дочерними процессами

Разработайте универсальную программу перенаправления стандартного ввода-вывода

**pipeInput.cpp**

#include <iostream>

#include <fstream>

#include <iostream>

#include <string>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <sys/utsname.h>

#include <ctime>

using namespace std;

#define FILENAME "myfile"

string executeCommand(const char\* cmd);

string getInfo();

int main()

{

char ch;

ofstream file(FILENAME);

if (!file.is\_open())

{

cerr << "Unable to open file\n";

return 1;

}

// Redirect stdout to the file

streambuf \*original\_stdout = cout.rdbuf(file.rdbuf());

cout << getInfo();

// Restore stdout

cout.rdbuf(original\_stdout);

file.close();

return 0;

}

string executeCommand(const char\* cmd)

{

string result = "";

FILE\* pipe = popen(cmd, "r");

if (!pipe)

{

cerr << "popen() failed\n";

return result;

}

char buffer[128];

while (fgets(buffer, sizeof(buffer), pipe) != nullptr)

{

result += buffer;

}

pclose(pipe);

return result;

}

string getInfo()

{

time\_t now = time(0);

tm\* localTime = localtime(&now);

char buffer[80];

strftime(buffer, sizeof(buffer), "%Y-%m-%d", localTime);

string date = buffer;

strftime(buffer, sizeof(buffer), "%H:%M:%S", localTime);

string time = buffer;

utsname unameData;

string osname = executeCommand("uname -a");

string osVersion = executeCommand("date");

string calendar = executeCommand("cal");

return "OS Name: " + osname+ "OS Version: " + osVersion

+ "Current Date and Time: " + date+" "+time + "\n"

+ "Calendar:\n" + calendar;

}

**pipeOutput.cpp**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

char ch;

ifstream file("myfile");

if (!file.is\_open()) {

cerr << "Unable to open file\n";

return 1;

}

// Redirect stdin to the file

streambuf \*original\_stdin = cin.rdbuf(file.rdbuf());

// Read from stdin and write to stdout

while (cin.get(ch)) {

cout.put(ch);

}

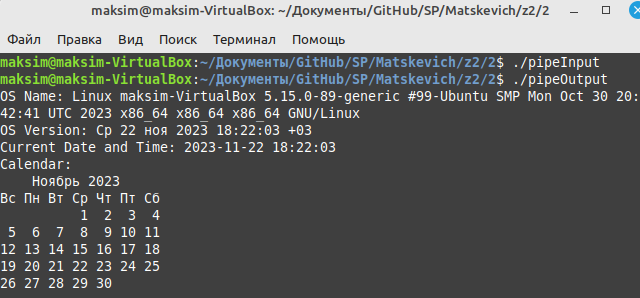
// Restore stdin

cin.rdbuf(original\_stdin);

file.close();

return 0;

}

Рис.4 Перенаправление между процессами

Выполните программу, указанную в упражнении 2, пример №3 с выводом со-

держимого каталогов на выбор: proc, lib, mnt, usr, etc, home или dev.

**main.cpp**

#include <iostream>

#include <cstring>

#include <unistd.h>

#include <sys/wait.h>

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

int pf[2];

pid\_t pid1, pid2;

if (argc < 3) {

std::cerr << "Too few arguments\n";

return 1;

}

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

std::cerr << "pipe() error\n";

return 1;

}

if ((pid1 = fork()) == 0) {

// Child process 1

dup2(pf[1], 1);

close(pf[0]);

execlp("ls", "ls", argv[1], nullptr);

std::cerr << "exec() [1] error\n";

return 1;

}

if ((pid2 = fork()) == 0) {

// Child process 2

dup2(pf[0], 0);

close(pf[1]);

execlp("grep", "grep", "-i", argv[2], nullptr);

std::cerr << "exec() [2] error\n";

return 1;

}

// Parent process

close(pf[0]);

close(pf[1]);

waitpid(pid1, nullptr, 0);

waitpid(pid2, nullptr, 0);

return 0;

}

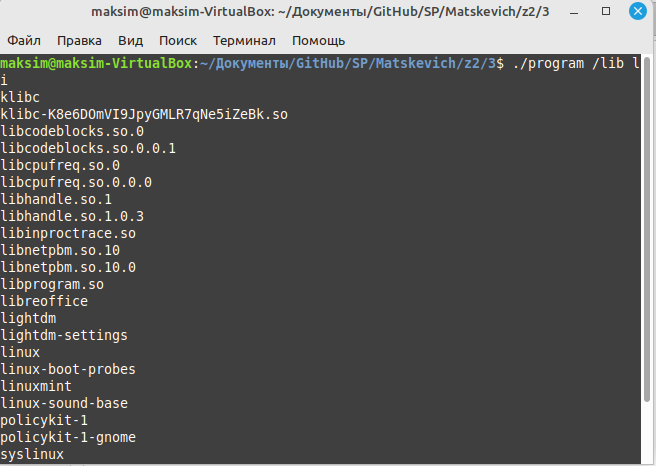


Рис.5 Содержимое каталогов

**Задание 3**

Разработайте универсальную программу передачи данных с помощью именованного канала

**Решение**

**client.cpp**

#include <iostream>

#include <fstream>

#include <cstring>

#include <fcntl.h>

#include <unistd.h>

using namespace std;

#define FIFO\_NAME "myfifo"

string executeCommand(const char\* cmd);

string getInfo();

int main() {

int fifo;

fifo = open(FIFO\_NAME, O\_WRONLY);

if (fifo == -1) {

cerr << "Cannot open fifo\n";

return 1;

}

string text = getInfo();

if (write(fifo, text.c\_str(), text.length()) == -1) {

cerr << "write() error\n";

return 1;

}

close(fifo);

return 0;

}

string executeCommand(const char\* cmd)

{

string result = "";

FILE\* pipe = popen(cmd, "r");

if (!pipe)

{

cerr << "popen() failed\n";

return result;

}

char buffer[128];

while (fgets(buffer, sizeof(buffer), pipe) != nullptr)

{

result += buffer;

}

pclose(pipe);

return result;

}

string getInfo()

{

time\_t now = time(0);

tm\* localTime = localtime(&now);

char buffer[80];

strftime(buffer, sizeof(buffer), "%Y-%m-%d", localTime);

string date = buffer;

strftime(buffer, sizeof(buffer), "%H:%M:%S", localTime);

string time = buffer;

string osname = executeCommand("uname -a");

string osVersion = executeCommand("date");

string calendar = executeCommand("cal");

return "OS Name: " + osname+ "OS Version: " + osVersion

+ "Current Date and Time: " + date+" "+time + "\n"

+ "Calendar:\n" + calendar;

}

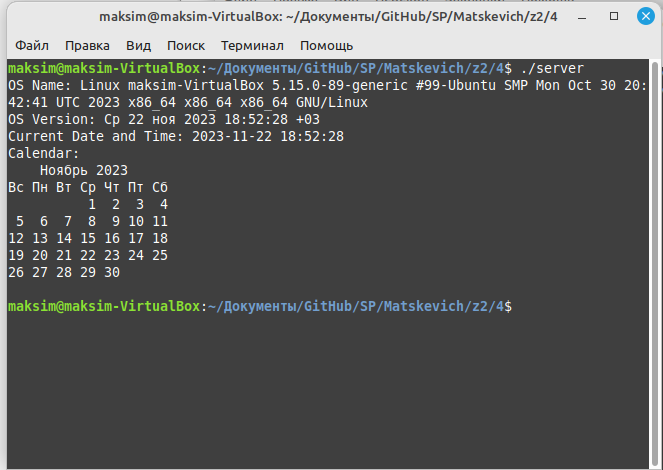


Рис.4 Передача через именованный канал

Вывод*:* В ходе выполнения лабораторной работы научились работать с каналами.

**Контрольные вопросы**

6. Команды, системные вызовы и библиотечные функции, относящиеся к каналам, и библиотеки их подключения.

Стандартные потоки:

std::cin: стандартный ввод (клавиатура).

* std::cout: стандартный вывод (экран).
* std::cerr и std::clog: стандартные потоки ошибок.

Функции работы с файлами:

* std::ifstream и std::ofstream: для чтения и записи файлов.
* std::fstream: комбинированный поток для ввода/вывода файлов.

Системные вызовы и функции POSIX:

* pipe(): создание канала для межпроцессного взаимодействия.
* fork(): создание нового процесса.
* close(): закрытие дескриптора файла.