Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное   
образовательное учреждение высшего образования   
«Самарский национальный исследовательский университет   
имени академика С.П. Королева»

Институт информатики, математики и электроники

Факультет информатики

Кафедра технической кибернетики

Отчет по лабораторной работе № 2

Дисциплина: «Операционные системы»

Тема «Операционная система Ubuntu: системные вызовы и работа с файлами»

Выполнил: Абрамов Данила Олегович

Группа: 6208-010302D

Самара, 2021

**Задание на лабораторную работу**

# Необходимо написать программу для работы с файлами, получающую информацию из командной строки или из консоли ввода.

**Текст программы**

#include <iostream>

#include <fstream>

#include <sys/stat.h>

#include <iomanip>

using namespace std;

void getHelp() {

cout << "cp - копирование файлов\n\n"

<< "mv - переместить файл\n\n"

<< "info - получить информацию о файле\n\n"

<< "quit - остановить программу\n\n"

<< "chmod - изменить права доступа\n\n"

<< "4000 - установить при выполнении идентификатор пользователя\n"

<< "2000 - установить при выполнении идентификатор группы\n"

<< "1000 - sticky бит\n"

<< "400 - владелец может читать\n"

<< "200 - владелец может писать\n"

<< "100 - владелец может выполнять файл или искать в каталоге\n"

<< "40 - группа-владелец может читать\n"

<< "20 - группа-владелец может писать\n"

<< "10 - группа-владелец может выполнять файл или искать в каталоге\n"

<< "4 - все остальные могут читать \n"

<< "2 - все остальные могут писать\n"

<< "1 - все остальные могут выполнять файл или искать в каталоге\n\n";

}

void cp() {

string fileName, copyName;

cin >> fileName >> copyName;

ifstream fin(fileName);

if (!fin) {

cout << "ERROR: Файл \"" << fileName << "\" не может быть открыт\n";

fin.close();

return;

}

if (fileName == copyName) {

cout << "ERROR: Имена двух файлов совпадают\n";

fin.close();

return;

}

ofstream fout(copyName);

for (char c; fin.get(c); fout << c);

fin.close();

fout.close();

cout << "Успешно скопировано\n";

}

void mv() {

string fileName, folderName;

cin >> fileName >> folderName;

system(("mv " + fileName + " " + folderName).c\_str());

}

void getInfo() {

string fileName;

struct stat sb;

cin >> fileName;

if (stat(fileName.c\_str(), &sb) == -1) perror("stat");

else {

cout << "тип файла: ";

switch (sb.st\_mode & S\_IFMT) {

case S\_IFBLK: cout << setw(57) << "блочное устройство\n"; break;

case S\_IFCHR: cout << setw(60) << "символьное устройство\n"; break;

case S\_IFDIR: cout << setw(47) << "каталог\n"; break;

case S\_IFIFO: cout << setw(44) << "FIFO\n"; break;

case S\_IFLNK: cout << setw(56) << "символьная ссылка\n"; break;

case S\_IFREG: cout << setw(51) << "обычный файл\n"; break;

case S\_IFSOCK: cout << setw(45) << "сокет\n"; break;

default: cout << setw(55) << "неизвестно\n"; break;

}

cout

<< "inode: " << setw(40) << (long)sb.st\_ino << endl

<< "Режим доступа: " << setw(40) << (unsigned long)sb.st\_mode << endl

<< "Кол-во ссылок: " << setw(40) << (long)sb.st\_nlink << endl

<< "Владелец: " << setw(25) << "UID = " << (long)sb.st\_uid << " GID = " << (long)sb.st\_gid << endl

<< "Предпоч. размер бл. в/в:" << setw(35) << (long)sb.st\_blksize << " байт\n"

<< "Размер файла: " << setw(40) << (long long)sb.st\_size << endl

<< "Выделено блоков: " << setw(40) << (long long)sb.st\_blocks << endl

<< "Посл. изм. состояния: " << setw(41) << ctime(&sb.st\_ctime)

<< "Посл. доступ к файлу: " << setw(41) << ctime(&sb.st\_atime)

<< "Посл.изм.файла: " << setw(41) << ctime(&sb.st\_mtime);

}

}

void chmod() {

string fileName;

int md;

cin >> fileName >> md;

switch (md) {

case 4000: chmod(fileName.c\_str(), S\_ISUID) != -1 ? cout << "Идентификатор пользователя установлен\n" : cout << "ERROR\n"; break;

case 2000: chmod(fileName.c\_str(), S\_ISGID) != -1 ? cout << "Идентификатор группы установлен\n" : cout << "ERROR\n"; break;

case 1000: chmod(fileName.c\_str(), S\_ISVTX) != -1 ? cout << "sticky бит\n" : cout << "ERROR\n"; break;

case 400: chmod(fileName.c\_str(), S\_IRUSR) != -1 ? cout << "Владелец может читать\n" : cout << "ERROR\n"; break;

case 200: chmod(fileName.c\_str(), S\_IWUSR) != -1 ? cout << "Владелец может писать\n" : cout << "ERROR\n"; break;

case 100: chmod(fileName.c\_str(), S\_IXUSR) != -1 ? cout << "Владелец может выполнять файл или искать в каталоге\n" : cout << "ERROR\n"; break;

case 40: chmod(fileName.c\_str(), S\_IRGRP) != -1 ? cout << "Группа-владелец может

case 20: chmod(fileName.c\_str(), S\_IWGRP) != -1 ? cout << "Группа-владелец может читать\n" : cout << "ERROR\n"; break;писать\n" : cout << "ERROR\n"; break;

case 10: chmod(fileName.c\_str(), S\_IXGRP) != -1 ? cout << "Группа-владелец может выполнять файл или искать в каталоге\n" : cout << "ERROR\n"; break;

case 4: chmod(fileName.c\_str(), S\_IROTH) != -1 ? cout << "Все остальные могут читать n" : cout << "ERROR\n"; break;

case 2: chmod(fileName.c\_str(), S\_IWOTH) != -1 ? cout << "Все остальные могут писать\n" : cout << "ERROR\n"; break;

case 1: chmod(fileName.c\_str(), S\_IXOTH) != -1 ? cout << "Все остальные могут выполнять файл или искать в каталоге\n" : cout << "ERROR\n"; break;

default: cout << "Некорректный ввод \n";

}

}

int main() {

string comand;

getHelp();

do {

cout << "~$ ";

cin >> comand;

if (comand == "--help") getHelp();

else if (comand == "cp") cp();

else if (comand == "mv") mv();

else if (comand == "info") getInfo();

else if (comand == "chmod") chmod();

else if (comand != "quit") cout << "Некорректная команда, введите --help\n";

} while (comand != "quit");

}