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 и кибернетики

Кафедра технической кибернетики

Отчет по лабораторной работе №2

Дисциплина: «Операционные системы»

Тема «Операционная система Linux: файлы и системные вызовы»

Выполнила: Иванова А.А.

Группа: 6206-010302D

Самара, 2024

**Задание на лабораторную работу**

Необходимо написать программу для работы с файлами, получающую информацию из командной строки или из консоли ввода.  
Программа должна корректно обрабатывать ключи и аргументы в случае ввода из командной строки, либо программа должна  
показывать список действий в случае ввода из консоли ввода.  
Программа должна иметь возможность осуществлять:  
- копирование файлов,  
- перемещение файлов,  
- получение информации о файле (права, размер, время изменения),  
- изменение прав на выбранный файл.  
35  
Просто вызвать функцию для копирования файла нельзя.  
Также программа должна иметь help для работы с ней, он должен  
вызываться при запуске программы с ключом --help.  
Копирование файла должно производиться при помощи команд блочного чтения и записи в файл. Размер буфера для чтения  
и записи должен быть больше единицы. Не допускается так же  
производить копирование файла при помощи однократной команды чтения и записи, так как при таком подходе предполагается,  
что оперативной памяти достаточно, чтобы прочитать весь файл  
одной командой. Это неверно, так как в общем случае размер файла может существенно превышать оперативную память и файл  
подкачки.  
Основные модули для работы с файлами:  
- #include <fcntl.h>  
- #include <sys/stat.h>  
- #include <unistd.h>  
Основные процедуры для работы с файлами (помощь по ним  
вызывается из командной строки командой $ man  
<имя\_команды>) :  
- Изменение текущей маски прав для программы:  
int umask(int newmask).  
- Открытие файла: int open(char\* pathname, int flags, mode\_t  
mode).  
- Закрытие файла: int close(int fd).  
- Чтение, запись: size\_t read(int fd, void \* buf, size\_t length),  
size\_t read(int fd, const void \* buf, size\_t length), size\_t write(int fd,  
void \* buf, size\_t length), size\_t write(int fd, const void \* buf, size\_t  
length).  
- Установка указателя чтения / записи: int lseek(int fd, off\_t offset, int whence); где whence: SEEKSET (начало файла), SEEK\_CUR  
(текущая позиция в файле), SEEKEND (конец файла).  
36  
- Сокращение файла: int truncate(const char \*pathname, size\_t  
length), int ftruncate(int fd, size\_t length).  
- Синхронизация файлов: int fsync(int fd), int fdatasync(int fd).  
- Запрос информации о файле:  
#include <sys/stat.h> ;  
int stat(const char \*pathname, struct stat \*statbuf);  
int lstat (const char \*pathname, struct stat \*statbuf);  
int fstat(int fd, struct stat \*statbuf).  
- Изменение прав доступа:  
int chmod(const char \*pathname, mode\_t mode);  
int fchmod(int fd, mode\_t mode).  
- Создание жестких ссылок:  
int link(const char \*origpath, const char \*newpath).  
- Создание символических ссылок:  
int symlink(const char \*origpath, const char \*newpath).  
- Удаление файла:  
int unlink(char \*pathname).  
- Переименование файлов:  
int rename(const char \*oldpath, const char \*newpath).

**Текст программы**

#include <fstream>

#include <string>

#include <iostream>

#include <fcntl.h>

#include <sys/stat.h>

#include <unistd.h>

using namespace std;

char valid\_continue() {

cout << "Do you want to continue? (y/n) >> ";

char answer;

cin >> answer;

answer = tolower(answer);

while (answer != 'y' && answer != 'n' || cin.peek() != '\n') {

cout << "I don't understand you, sorry. Please, try again.\n";

cin.clear();

cin.ignore(10000, '\n');

cout << "\nDo you want to continue? (y/n) >> ";

cin >> answer;

}

return answer;

}

string error() {

string com;

cin >> com;

while (com != "cp" && com != "mv" && com != "ls" && com != "chmod" || cin.peek() != '\n') {

cout << "I don't understand you, sorry. Please, try again.\n";

cin.clear();

cin.ignore(10000, '\n');

cin >> com;

}

return com;

}

void defects() {

cout << "Error\n";

exit(0);

}

void copy(string filename, string newFilename) //cp копирование

{

if (filename == newFilename) defects();

ifstream fin;

size\_t bufsize = 4;

char\* buf = new char[bufsize];

fin.open(filename, ios::binary);

if (fin.is\_open()) {

ofstream fout;

fout.open(newFilename, ios::binary);

while(!fin.eof()) {

fin.read(buf, bufsize);

if (fin.gcount()) fout.write(buf, fin.gcount());

}

cout << "The copying was successful" << endl;

delete[] buf;

fin.close();

fout.close();

}

else {

defects();

return;

}

}

void move(string filename, string newfilename) { //mv перемещение

if(filename == newfilename) { defects();}

else rename(filename.c\_str(), newfilename.c\_str());

cout << "The file has been moved\n";

}

string permissions(int perm) { //параметры доступа

char permis[10] = "rwxrwxrwx";

char res[10];

int i, j;

for (i = 0, j = 256; i < 9; i++, j >>= 1)

res[i] = (perm & j) ? permis[i] : '-';

res[9] = '\0';

return res;

}

void info(string filename) // ls

{

struct stat statbu;

if(stat(filename.c\_str(), &statbu)) { defects();}

cout << permissions(statbu.st\_mode) << " " << statbu.st\_size << " " << statbu.st\_ctime << endl;

}

int toDec(int num) {

int d1 = num%10;//последнее

int d2 = (num/10) % 10; //среднее

int d3 = num/100;// левое

int res = d1 + d2\*8 + d3\*64;

return res;

}

void changePer(string filename, int permissions) {

if (chmod(filename.c\_str(), permissions)) {

defects();

}

cout << "Permissions changed successfully!" << endl;

}

void help(){

cout << "cp file1 file2 - copy file1 to file2\n";

cout << "mv file1 file2 - move file1 to file2\n";

cout << "ls file - information about the file\n";

cout << "chmod num file - change file permissions\n"; // —help

}

int main() {

char ans;

help();

cout << "Choosing an action: \n" << endl;

do {

string action, file1, file2;

int perm;

action = error();

if(action == "cp"){

cin >> file1 >> file2;

copy(file1, file2);

}

else if(action == "mv"){

cin >> file1 >> file2;

move(file1, file2);

}

else if(action == "ls"){

cin >> file1;

info(file1);

}

else if(action == "chmod"){

cin >> perm >> file1;

changePer(file1, toDec(perm));

}

ans = valid\_continue();

} while(ans == 'y');

return 0;

}