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 Ubuntu: системные вызовы и работа с файлами»

Выполнил: Белов А. А.

Группа: 6205-010302D

Самара, 2023

**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.

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

28

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:

o size\_t read(int fd, void \* buf, size\_t length);

o size\_t read(int fd, const void \* buf, size\_t length);

o size\_t write(int fd, void \* buf, size\_t length);

o 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:

o SEEKSET - Начало файла.

o SEEK\_CUR - Текущая позиция в файле.

o SEEKEND - Конец файла.

 Сокращение файла:

o int truncate(const char \*pathname, size\_t length);

o int ftruncate(int fd, size\_t length);

 Синхронизация файлов:

o int fsync(int fd);

o 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);

29

 Изменение прав доступа:

o int chmod(const char \*pathname, mode\_t mode);

o 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**

#//define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <fcntl.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <fstream>

#include <vector>

#include <map>

#include <cstdlib>

#include <string>

#include <sstream>

#include <ctime>

#include <unistd.h> //не знаею, походу на window не работает это

//#include <io.h>

#include <set>

#include <stdio.h>

//#include <locale.h>

#include <cmath>

using namespace std;

void skibidiDopDopDopYesYesYesWWW(int ss10){

int ss8 = 0;

int count = 0;

for (int i = 0; i < 3; ++i){

ss8 += ss10 % 8 \* pow(10,count++);

ss10 /= 8;

}

cout <<"Access rights: " << ss8 << endl;

}

// получаем информацию о файле

void get\_information(string comand ,const string& file\_name) {

int file\_descriptor = open(file\_name.c\_str(), O\_RDONLY); // открываем файл в ubuntu: open(file\_name.c\_str(), O\_RDONLY) // O\_RDONLY - определяет флаг доступа к файлу в режиме "только для чтения" (read-only). Это означает, что файл может быть только прочитан, но не изменен или записан.

if (file\_descriptor == -1) {// выходим из функции, при неудачном чтении файла

cout << "File open error! " << endl;

}

struct stat file\_info; // содержит информацию о файле, такую как размер файла...

if(fstat(file\_descriptor, &file\_info) == 0 || stat(file\_name.c\_str(), &file\_info) == 0 || lstat(file\_name.c\_str(), &file\_info) == 0) {

if (comand == "size")

cout << "File size: " << file\_info.st\_size << " byte" << endl;

if (comand == "right")

skibidiDopDopDopYesYesYesWWW(file\_info.st\_mode);

if (comand == "time\_last\_change")

cout << "Last modified time: " << ctime(&file\_info.st\_mtime) << endl;

if (comand == "device")

cout << "Device ID: " << file\_info.st\_dev << endl;

if (comand == "count\_blocks")

cout << "Count of blocks : " << file\_info.st\_blocks << endl;

if (comand == "rd")

cout << "size block input-output : " << file\_info.st\_blksize << endl;

close(file\_descriptor);

}

else

cout << "Error geting file information" << endl;

}

/\*

первая цифра(бит) - разрешение для владельца

вторая цифра(бит) - разрешение для группы

третья цифра(бит) - разрешение для всех остальных

\*/

/\*

0 - никаких прав;

1 - только выполнение;

2 - только запись;

3 - выполнение и запись;

4 - только чтение;

5 - чтение и выполнение;

6 - чтение и запись;

7 - чтение запись и выполнение.

\*/

int translate(int ss\_8){

int ss\_10 = 0;

int count = 0;

while (ss\_8){

ss\_10 += ss\_8 % 10 \* pow(8, count++);

ss\_8 /= 10;

}

//cout << ss\_10 << endl;

return ss\_10;

}

bool control\_input\_chmod(const string& comand) {

int num = stoi(comand);

while (num > 0) {

if (num % 10 >= 0 && num % 10 < 8)

num = num / 10;

else {

cout << "Wrong mode: " + comand << endl;

return 0;

}

}

return 1;

}

bool copy\_file(string old\_file, string new\_file){

int old\_d = open(old\_file.c\_str(), O\_RDONLY);

if (old\_d == -1){

cout << "File " << old\_file << "could not be opened " << endl;

return 0;

}

mode\_t per = S\_IRUSR | S\_IWUSR | S\_IRGRP | S\_IROTH; // модификаторы доступа 644

int new\_d = open(new\_file.c\_str(), O\_CREAT | O\_WRONLY , per); // созадали файл(если отсутствовал) и указали флаг "только для записи"

if (lseek(old\_d, 0, SEEK\_SET) == -1){

close(old\_d);

close(new\_d);

return 0;

}

char buffer[1024]; // буфер размером 1 КБ

ssize\_t bytes\_read;

while((bytes\_read = read(old\_d, buffer, 1024)) > 0){

ssize\_t bytes\_write = 0;

do {

ssize\_t result = write(new\_d, buffer + bytes\_write, bytes\_read - bytes\_write);

if (result == -1){

close(old\_d);

close(new\_d);

return 0;

}

bytes\_write += result;

} while(bytes\_write < bytes\_read);

}

if (bytes\_read == -1){

close(old\_d);

close(new\_d);

return 0;

}

close(old\_d);

close(new\_d);

return 1;

}

void copy\_help(){

cout << "copy 'enter the source path of the file' 'enter final path' " << endl;

}

void move\_help(){

cout << "move 'enter the source path of the file' 'enter final path' " << endl;

cout << "move 'old\_filaname.txt' 'new\_fulename.txt' " << endl;

}

void get\_help(){

cout << " get size filename.txt" << endl;

cout << " get right filename.txt" << endl;

cout << " get time\_last\_change filename.txt" << endl;

}

void chmod\_help(){

cout << " chmode 'code rights' filename.txt" << endl;

}

int main() {

//string end; // перемнная для выхода из программы

do {

string str; // сюда будет записана строка, введённая с клавиатуры

getline(cin, str); // считывания всей строки

stringstream ss(str); // создаём поток ввода из строки

vector<string> comands; // создаём вектор в который будем добавлять слова

string words; // создаём переменную в которой будет храниться слово

while (ss >> words) // используем потоковый ввод для разбиения строки на слова

comands.push\_back(words); // добавляем слово в вектор

//for (int i = 0; i < comands.size(); ++i) cout << comands[i] << " "; // выводим все слова в векторе

// Проверяем есть ли ключ "--help" и выполняем команду с ключом

if (comands.size() == 2 && comands[1] == "--help") {

map<string, string> double\_string;

string key = comands[1]; // = "--help"

string value = comands[0]; // = команда (rename, )

double\_string[key] = value;

// выводим ключ и его значение

for (auto i : double\_string) {

cout << "-------------------------------" << endl;

cout << "check key-value: " << endl;

cout << "Key: " << i.first << endl;

cout << "Value: " << i.second << endl;

}

if (comands[0] == "copy")

copy\_help();

else if (comands[0] == "move")

move\_help();

else if (comands[0] == "get")

get\_help();

else if (comands[0] == "chmod")

chmod\_help();

cout << endl;

}

// выполняем команды терминала

if (comands.size() == 3) {

if (comands[0] == "move" ) {

if (!rename(comands[1].c\_str(), comands[2].c\_str()))

cout << "File " << comands[1] << "moved successfully " << endl;

else

cout << "Fail to moved " << endl;

}

if (comands[0] == "copy" ) {

if (copy\_file(comands[1], comands[2]))

cout << "File " << comands[1] << " " << comands[0] << " successfully to " << comands[2] << endl;

else

cout << "Fail" << endl;

}

if (comands[0] == "get") {

if (comands[1] == "size" || comands[1] == "right" || comands[1] == "time\_last\_change" || comands[1] == "device" || comands[1] == "count\_blocks" || comands[1] == "rd")

get\_information(comands[1], comands[2]);

else

cout << "Error" << endl << "use --help " << endl;

}

if (comands[0] == "chmod") {

//cout << "надо так: chmod 777 file.txt" << endl;

if (control\_input\_chmod(comands[1])){ // проверка ввода для кода доступа

int num = translate(stoi(comands[1])); // переводим из 8-ричной сс в 10-сс

chmod(comands[2].c\_str(), num); // выполяем изменение доступа

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

}

else

cout << "Error acces changed! " << endl;

}

}

if (comands.size() == 1 && comands[0] == "end") // выход из бесконечного цикла

return 0;

} while (true);

}