Министерство науки и образования РФ  
Новосибирский государственный технический университет  
Кафедра ТПИ

Лабораторная работа №3

по дисциплине «Управление ресурсами в вычислительных системах»

Факультет: ПМИ

Группа: ПМИ-31

Студенты: Суслов А.В.

Эльвейн Н.Л.  
 Олимпиади М.Б.

Преподаватель: Хайленко Е.А.  
 Филиппова Е.В.

Вариант: 1

1. **Цель работы**

Ознакомиться с системой управления вводом-выводом в ОС UNIX и основными структурами данных, используемыми этой системой. Исследовать механизм работы системы управления вводом-выводом.

1. **Задание**

Вариант №1: Процесс открывает N файлов, реально существующих на диске, либо вновь созданных. Разработать программу, демонстрирующую динамику формирования таблицы описателей файлов и изменения информации в ее элементах (при изменении информации в файлах).

1. **Описание метода решения задачи**

Заполняем таблицу описателей файлов начальными данными для каждого из имени файлов.  
Последовательно выполняем действия сценария, после каждого действия обновляем таблицу описателей файлов с помощью созданной для этого функции.

1. **Описание разработанного программного средства**

Программа написана на языке C. Для получения исполняемого файла исходный текст программы следует скомпилировать каким-либо компилятором языка C. Исполняемый файл может работать под любой UNIX-совместимой ОС, поддерживающей соответствующий формат исполняемого файла.

Запуск программы осуществляется командой:

./main

1. **Текст программы**

#include <limits.h>

#define FNAME\_MAX (NAME\_MAX+1)

// create three files

const char names[][FNAME\_MAX] = {

"file1.dat",

"file2.inf",

"file3.txt"

};

#include <stdio.h>

#include <sys/stat.h>

#include <sys/sendfile.h>

#include <unistd.h>

#pragma region system(pause) Unix-implementation

#ifndef WIN32

#include <termios.h>

void systempause() {

struct termios info, info\_prev;

tcgetattr(0, &info); /\* get current terminal attirbutes; 0 is stdin \*/

tcgetattr(0, &info\_prev);

info.c\_lflag &= ~ICANON; /\* disable canonical mode \*/

info.c\_cc[VMIN] = 1; /\* wait until at least one keystroke available \*/

info.c\_cc[VTIME] = 0; /\* no timeout \*/

tcsetattr(0, TCSANOW, &info); /\* set immediately \*/

printf("Press any key to continue...\n");

getchar();

tcsetattr(0, TCSANOW, &info\_prev); /\* set buffered \*/

}

#else

#include <stdlib.h>

void systempause() {

system("pause");

}

#endif

#pragma endregion

#include <string>

#include <vector>

using namespace std;

/\* Структура таблицы открытых файлов в программах должна иметь вид:

filename, дескриптор файла, ссылка на таблицу файлов (opt)

\*/

struct file\_d\_tr {

string filename;

FILE \*stream;

int fd;

int inode;

int locate;

int size;

file\_d\_tr(string filename) {

this->filename = filename;

stream = NULL;

};

};

extern const unsigned int FILE\_COUNT = sizeof(names) / FNAME\_MAX;

/\* Updates existing table of currently opened files. \*/

void update\_table\_d(vector<file\_d\_tr> &table) {

struct stat buf;

for (int i = 0; i < table.size(); i++) {

if (table[i].stream != NULL) {

int ret = fstat(table[i].fd, &buf);

table[i].inode = buf.st\_ino;

table[i].locate = ftell(table[i].stream);

table[i].fd = fileno(table[i].stream);

table[i].size = buf.st\_size;

}

}

}

/\* Prints table of currently opened files. \*/

void print\_table\_d(vector<file\_d\_tr> table, FILE \*f = stdout) {

fprintf(f, "File name\tDescriptor\tinode\tlocate\tsize\t\n");

for (int i = 0; i < table.size(); i++) {

if (table[i].stream != NULL) {

fprintf(f, "%s\t%d\t\t%d\t%d\t%d\n", table[i].filename.c\_str(), table[i].fd, table[i].inode, table[i].locate, table[i].size);

}

}

}

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

vector<file\_d\_tr> table\_d; // таблица дескрипторов

FILE \*f;

for (int i = 0; i < FILE\_COUNT; i++) {

fclose(fopen(names[i], "w"));

table\_d.push\_back(\*(new struct file\_d\_tr(names[i])));

}

// build open files table

print\_table\_d(table\_d);

systempause();

printf("Opening 1st file.\n");

table\_d[0].stream = fopen(table\_d[0].filename.c\_str(), "r");

update\_table\_d(table\_d);

print\_table\_d(table\_d);

systempause();

printf("Opening 2nd file.\n");

table\_d[1].stream = fopen(table\_d[1].filename.c\_str(), "r");

update\_table\_d(table\_d);

print\_table\_d(table\_d);

systempause();

printf("Opening 3rd file.\n");

table\_d[2].stream = fopen(table\_d[2].filename.c\_str(), "w");

update\_table\_d(table\_d);

print\_table\_d(table\_d);

systempause();

printf("Truncating 3rd file.\n");

ftruncate(table\_d[2].fd, 0);

update\_table\_d(table\_d);

print\_table\_d(table\_d);

systempause();

printf("Copying content of 2nd file to 3rd file.\n");

off\_t offset = 0;

sendfile(table\_d[2].fd, table\_d[1].fd, &offset, table\_d[1].size);

update\_table\_d(table\_d);

print\_table\_d(table\_d);

systempause();

printf("=== END OF SCENARIO ===\n");

systempause();

return 0;

}

1. **Результат работы программы**

