Министерство науки и образования РФ

ФГБОУ ВО «Омский государственный технический университет»

Кафедра «Информатика и вычислительная техника»

**ОТЧЕТ**

**по выполнению лабораторной работы**

**по операционным системам**

**в ходе дистанционной формы обучения**

**Номер 2**

Выполнил: студент гр.

Омск – 2020

# **Цель работы.**

Изучение связи файлов, их хэндлов и управляющих блоков файлов.

# **Задание.**

Результат выполнения лабораторной работы должен состоять из двух программ для Linux. Первая программа должна создавать текстовый файл, вводя данные со стандартного ввода. (Более детально: открывает файл для записи, читает текст со стандартного ввода и выводит этот прочитанный текст в файл.) Вторая программа открывает тот же файл (созданный перед этим другой программой) для чтения и хэндл, полученный при этом открытии, запоминает в 1-й переменной для хэндла. Используя этот хэндл, далее с помощью функции dup() получается новое значение хэндла для доступа к тому же файлу (2-й хэндл). Еще раз открывается тот же файл, запоминая 3-е значение хэндла. С помощью первого хэндла программа позиционирует чтение для 10-й позиции файла от начала этого файла. Далее программа должна выводить числовые значения всех трех хэндлов на экран. Используя по очереди все 3 хэндла, из файла читаются по 7 символов и тут же эти три прочитанных текста выводятся на экран, каждая в своей строке. Результаты вывода объяснить.(Базовый вариант).

**Краткие теоретические сведения.**

При работе программы с файлом нужно где-то поддерживать рабочие параметры этой работы. Особенно очевидна необходимость вести учет до какого места в файле он прочитан.

Более-менее совершенные реализации инструментов работы с файловой системой стали включать специальные структуры данных для работы с последовательно используемым файлом, которые в свое время получили название file control block – управляющий блок файла. В данном контексте слово control в одном из его значений на английском языке обозначает именно контроль, контролирование и учет, а не собственно действия по принудительному управлению.

В блоке управления файлом важнейшим элементом (полем структуры данных) является именно порядковый номер байта в файле, с которого реально будет выполняться следующая по времени операция чтения или записи. Эта величина называется текущей позицией ввода-вывода в файле (и иногда называется смещением – offset). Для того, чтобы начать читать (или писать внутри уже записанного файла) не с достигнутого ранее места, достаточно принудительно установить эту позицию с помощью специально предназначенных для этого функций. В ОС Linux основной из этих функций является функция lseek. Функция имеет три аргумента, в первом указывается файловый дескриптор (хэндл), второй задает величину используемого в ней смещения, последний обозначает точку отсчета для этого смещения – от начала файла, от предыдущего позиции или от конца файла.

Имея уже открытый файл, а следовательно введенный в использование его управляющий блок, можно получить другой номер файлового дескриптора, которым можно пользоваться как средством доступа к тому же управляющему блоку, а следовательно, и к тому же файлу.

Для этого в Unix/Linux служат системные функции dup и dup2. Они создают действующую файловый дескриптор к тому же файлу, точнее к тому же управляющему блоку файла. Функция dup() просто предоставляет другой номер для доступа к тому же управляющему блоку. Причем, как второстепенное свойство, ОС выделяет для этого минимально доступный (не занятый) номер. После такого действия операции с файлом могут осуществляться как с помощью старого, так и нового файлового дескриптора.

Функция dup2 с прототипом int dup2(int old\_handle, int new\_handle) предоставляет возможность использовать желаемый номер в качестве файлового дескриптора уже открытого файла. Именно, первый параметр этой функции задает действующий файловый дескриптор, а второй параметр должен быть численно равен желаемому номеру. Обе функции при невозможности выполнения возвращают служебное значение, равное -1, что является признаком ошибки.

**Код программы:**

*Файл lab\_two.c:*

#include <stdio.h>

#include <fcntl.h>

int main(){

char char\_massive[50]; // массив для хранения текста

int kolch; // целое число являющеся количеством прочитанных байтов

int fhandle; // хэндл файла

/\*

Открытие файла

O\_WRONLY - Открываем для записи

O\_CREAT - Создаем файл если он не существует

O\_TRUNC - Если файл есть, то обнуляем данные в нем

0777 - Каждый может читать, писать и выполнять файл

\*/

fhandle=open("file", O\_WRONLY | O\_CREAT | O\_TRUNC, 0777);

/\*

записываем в массив строку и выводим с помощью функции write

\*/

kolch = sprintf(char\_massive,"Vvedite text dly zapisi v file:\n");

write(1, char\_massive, kolch);

/\*

Читаем с пользовательского ввода

Возвращает реальное количество прочитаных байт

\*/

kolch=read(0, char\_massive, 50);

/\*

запись в файлы

\*/

write(fhandle, char\_massive, kolch);

close(fhandle); // закрытие файла

return 0;

}

*Файл lab\_two\_c.c:*

#include <stdio.h>

#include <fcntl.h>

void main(){

int fhandle1, // хэндлы

fhandle2,

fhandle3,

kolch; // количество прочитанных байтов

char char\_massive[50]; // массив символов для хранения текста

fhandle1 = open("file", O\_RDONLY); // открытие файла для чтения и передача хэндла переменной fhandle1

fhandle2 = dup(fhandle1); // предоставляем доступ к блоку где обрабатывается файл, fhandle1

fhandle3 = open("file", O\_RDONLY); // открытие файла для чтения и передача хэндла переменной fhandle3

// запись в массив хэндлов и вывод их поочередно

kolch = sprintf(char\_massive, "%d handle1\n", fhandle1);

write(1, char\_massive, kolch);

kolch = sprintf(char\_massive,"%d handle2\n", fhandle2);

write(1, char\_massive,kolch);

kolch = sprintf(char\_massive,"%d handle3\n", fhandle3);

write(1, char\_massive, kolch);

/\*

lseek. Функция имеет три аргумента, в первом указывается файловый дескриптор (хэндл),

второй задает величину используемого в ней смещения, последний обозначает точку отсчета

для этого смещения – от начала файла, от предыдущего позиции или от конца

файла.

Образуется два управляющих блока. В первом 1 и 2 хэндл, во втором 3. Что объясняется результат.

SEEK\_SET - Смещение отсчитывается от начала файла

\*/

lseek(fhandle2, 10, SEEK\_SET);

// чтение с файла 7 символов - хэндл 1

kolch=read(fhandle1, char\_massive, 7);

// вывод на экран консоли прочитанное

write(1, char\_massive, kolch);

// чтение с файла 7 символов - хэндл 2

kolch=read(fhandle2, char\_massive, 7);

// вывод на экран консоли прочитанное

write(1, char\_massive, kolch);

// чтение с файла 7 символов - хэндл 3

kolch=read(fhandle3, char\_massive, 7);

// вывод на экран консоли прочитанное

write(1, char\_massive, kolch);

}

}

# **Основные сложности, обнаруженные в ходе выполнения работы:-**

**Скриншоты выполнения программы:**

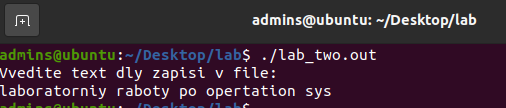


Рисунок 1. Выполненная программа 1

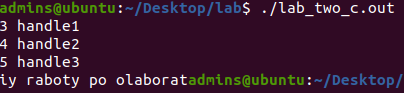


Рисунок 2. Выполненная программа 2

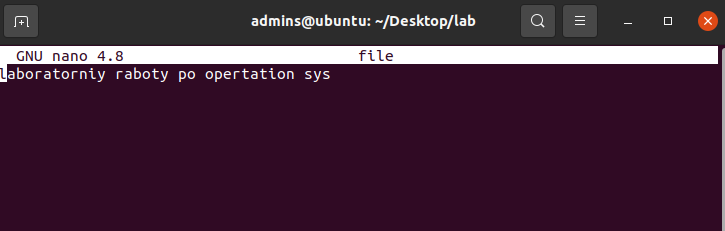


Рисунок 3. Результат первой программы

**Краткое комментирование тех программных конструкций, которые требовали переделок:-**

**Краткие выводы:**

Разработаны программы на Linux, реализующие создание файла с запись в него строки и чтением с файла. Имеются три хэндла, первый отрывает файл, второй через функцию дублирует, тем самым включившись в управляющий блок и способный менять его данные, а третий открывает файл заново. Происходит смещение на 10 символов и чтение по 7 символов из файла, который был создан в первой программой. Второй дескриптор имеет доступ ко первому блоку и смещая на 10 символов вперед, происходит смещение и в первом т.к. они являются одним блоком.

**Использованная литература**

1. Флоренсов А. Н. Ф 73 Операционные системы для программиста. Учеб. пособие. Омск: Изд-во ОмГТУ, 2005. -240 с.
2. Операционные системы: Учебное пособие / Ю.В. Марапулец. – Петропавловск-Камчатский: КамчатГТУ, 2008 – 235 с.