МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт кибернетики
Базовая кафедра №252 – информационной безопасности

КУРСОВАЯ РАБОТА

По дисциплине «Языки программирования»

Тема курсовой работы: «Виды файлов. Операции над файлами. Файловый ввод-вывод в языке C»

Студент группы ККСО-03-19 Николенко В.О. _____ *(подпись)*

Руководитель курсовой работы профессор, д.т.н. Лукинова О.В _____ *(подпись)*

Работа представлена к защите «__» _____ 2020 г.

Допущен к защите «__» _____ 2020 г.

Оценка «_____»

Москва – 2020

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

Ответственные за сохранное расположение информационных материалов отвечают запоминающие устройства. Для их успешного и безошибочного функционирования необходимо обязательное наличие программного интерфейса, структурирующего расположение любой информации, и предоставляющего упорядоченные способы управления доступными ресурсами. Такой урегулированный контролируемый способ внутренней организации, расположения и упорядочивания данных, в соответствии с собственными методами каталогизации и озаглавливания, на различных носителях информации в компьютерах и ноутбуках, а также в разнообразных сторонних электронных устройствах, получил обобщающее название файловая система.

Файловая система является важной частью любого накопителя информации. Она позволяет организовывать файловое пространство и работать с ней операционной системе.

Все это нужно, чтобы мы могли быстро получать доступ к своим файлам, записывать новые, и вообще взаимодействовать со своим накопителем информации.

2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Файл – именованная область внешней памяти, выделенная для хранения массива данных. Данные, содержащиеся в файлах, имеют самый разнообразный характер: программы на алгоритмическом или машинном языке; исходные данные для работы программ или результаты выполнения программ; произвольные тексты; графические изображения и т. п.

2.1. Описание устройства файловой системы в ОС Windows.

Файловой системой называется функциональная часть операционной системы, обеспечивающая выполнение операций над файлами. Примерами файловых систем являются FAT (существуют три основные версии FAT: FAT12, FAT16 и FAT32) и NTFS.

Рассмотрим сравнительную таблицу для этих файловых систем:

Таблица 1. Сравнение FAT, FAT32 и NTFS.

Файловая система	Параметры Разделы тома	Максимальный размер файла
Fat	От 1.44 МБ до 4 ГБ	2 гб
Fat 32	Теоретически возможен размер тома от 512 МБ до 2 Тбайт. Сжатие не поддерживается на уровне файловой системы	4 гб
NTFS	Минимальный рекомендуемый размер составляет 1,44 МБ, а максимальный - 2 Тбайт. Поддержка сжатия на уровне файловой системы для файлов, каталогов и томов.	Максимальный размер ограничен лишь размером тома (Теоретически - 264 байт минус 1 килобайт. Практически - 244 байт минус 64 килобайта)

Операционная система Windows 8, Windows 8.1 поддерживает несколько файловых систем: NTFS, FAT и FAT32. Но работать может только на NTFS, то есть установлена может быть только на раздел жесткого диска, отформатированного в данной файловой системе. Обусловлено это теми особенностями и инструментами безопасности, которые предусмотрены в NTFS, но отсутствуют в файловых системах Windows предыдущего поколения: FAT16

и FAT32.

Преимущества NTFS: реализация повышенных требований безопасности, производительности, надежности и эффективности работы с данными (файлами) на диске. Так, одной из основных целей создания NTFS было обеспечение скоростного выполнения операций над файлами (копирование, чтение, удаление, запись), а также предоставление дополнительных возможностей: сжатие данных, восстановление поврежденных файлов системы на больших дисках и т.д.

2.2. Описание устройства файловой системы в ОС Linux.

Операционные системы хранят данные на диске при помощи файловых систем. Классическая файловая система представляет данные в виде вложенных друг в друга каталогов (их ещё называют папками), в которых содержатся файлы. Каталог, в котором содержатся все остальные каталоги и файлы называется корневым.

Если жёсткий диск разбит на разделы, то на каждом разделе организуется отдельная файловая система с собственным корнем и структурой каталогов.

В Linux путь к корневому каталогу — "/". Полные имена (пути) всех остальных каталогов получаются из "/ к которому дописываются справа имена последовательно вложенных друг в друга каталогов. Имена каталогов в пути также разделяются символом "/"("слэш").

Библиотеки

Как правило, для корректной работы большинства программ необходимы зависимости, которые называют библиотеками – набором файлов, к которым подключается программа.

Монтирование

Монтирование — это подключение в один из каталогов целой файловой системы, которая находится где-то на другом устройстве.

Структура файловой системы в ОС Linux.

Файловая система Linux достаточно сильно отличается от Windows. Сама файловая система, структура каталогов, размещение конфигурационных, исполняемых и временных файлов - особенности данной ОС. В отличие от

Windows, программа не находится в одной папке, а, как правило, распределена по корневой файловой системе.

2.3. Описание форматов имён файлов.

В программе объявляется или принимается по умолчанию (INPUT и OUTPUT) имя логического файла, которое используется при обращении к процедурам и функциям работы с файлом. Для обработки данных физических файлов, имя логического файла должно быть связано с именем физического файла с помощью процедуры ASSIGN.

Имя физического файла, который связывается с файлом программы, может быть определено с помощью любого выражения строкового типа. Результатом выполнения выражения должно быть имя файла, допустимое в DOS:

• оно может содержать до 255 разрешенных символов: латинских букв, цифр и символов: !, #, $, %, &, ', (,), _;

• имя может начинаться с любого разрешенного символа;

• за именем может следовать точка и расширение - последовательность от одного до трех разрешенных символов.

Перед именем файла может быть путь к файлу - имя МД и (или) имя текущего каталога и имена каталогов вышестоящего уровня. Максимальная длина имени вместе с путем 79 символов. Путь может формироваться в процессе выполнения программы.

Таблица 2. Расширения файлов.

Расширение	Тип файла
doc,txt,rtf	Текстовые файлы
exe	Исполняемая программа
wav,mid,mp3	Звуковые файлы
avi,wmf,mp4	Видеофайлы
ppt,pps	Файлы презентации
zip,7z,rar	Архивы
htm,html	Web-страницы
bmp,gif,jpg,png	Графические файлы
bin	Двоичный файл

2.4. Классификацию типов файлов по разным основаниям.

Классификация по организации доступа к файлу.

Файл с последовательным доступом рассматривается как последовательность значений, которые передаются в порядке их поступления (от программы или из окружения). Если файл открыт, то передача начинается с начала файла. Поиск в таких файлах осуществляется последовательным считыванием файла с начала и сравнением «всего» с искомым. Так же и обращение к определённому участку файла каждый раз требует «чтения с начала». Примером последовательных файлов являются текстовые файлы (*.txt).

В файлах прямого доступа есть возможность изменять порядок получения данных, управляя позицией, в которой выполняется ввод-вывод. Эта позиция определяется индексом (аналогом адреса внутренней памяти). Файлы с прямым доступом — файлы, хранящие информацию в структурированном (для поиска и обращения) виде. Поиск в таких файлах осуществляется в области адресов (ключей) и завершается обращением непосредственно к искомому участку. Если ключ реализован в виде целого числа, то он очень похож на обычный индекс, используемый для обозначения компонентов массива.

Классификация по типу файла.

Текстовый файл специализируется на хранении текстовой информации, представленной в символах ASCII. Как правило, такие файлы снабжаются специфичным только для них расширением .txt и могут быть открыты любым текстовым редактором.

Двоичный (бинарный) файл — файлы, в которых данные представлены в двоичной системе счисления, в широком смысле: последовательность произвольных байтов. Название связано с тем, что байты состоят из бит, то есть двоичных цифр.

Типизированный файл — это файл, все компоненты которого одного типа, заданного при объявлении файловой переменной. В отличие от узкоспециализированного текстового файла, типизированный файл предназначен для работы с данными, определяемыми программистом. В качестве них могут выступать как любые целые, вещественные, символьные, логические и строковые типы данных, так и записи, состоящие из только что перечисленных типов.

Специальные файлы – это файлы, которые содержат в себе указа-

тель на драйвер для работы с определённым физическим устройством компьютера. Операционная система использует их для обращения и работы с физическим устройством. В ОС Linux они хранятся в каталоге /dev, а в ОС Windows они преимущественно находятся в папке Windows корневого каталога.

2.5. Описание атрибутов файлов.

Атрибут файла — это некая метка (или флаг), которая сообщает операционной системе о существовании особых правил для работы с конкретным файлом.

Атрибуты для Linux

Существует две команды для управления атрибутами файла: lsattr(1) и chattr(1). Команда lsattr служит для просмотра атрибутов, ну а команда chattr служит для изменения атрибутов. Атрибуты могут быть установлены только для каталогов и обыкновенных файлов. Доступны следующие атрибуты:

• A (no Access time) - при обращении к этому файлу (для чтения или записи), у файла не будет модифицироваться время последнего обращения.

• a (append only) - это добавление в конец файла (append - единственная доступная операция записи). В случае каталога это означает, что вы можете только добавлять файлы, но не можете переименовывать или удалять любой существующий файл внутри этого каталога.

• d (no dump) - создает резервную копию любой файловой системы, у которой в файле /etc/fstab значение dump counter установлено в 1.

• i (immutable) - файл или каталог с таким атрибутом не может изменяться вообще. Также такой файл не может быть удален. Этот атрибут также предотвращает изменения времени доступа, поэтому вы не должны совместно использовать атрибуты A и i.

• s (secure deletion) - при удалении файла, помеченного таким атрибутом, место на диске, которое занимал файл, будет заполнено нулями.

• S (Synchronous mode) - если установлен такой атрибут, то все изменения в файл будут записаны немедленно. Попросту говоря, этот атрибут снимает буферизацию записи для этого файла.

Атрибуты для Windows

В ОС Windows существует 4 возможных атрибута:

• Read-only (только для чтения) – файлы с этим атрибутом не могут быть изменены ни пользователем, ни какой-либо программой, их можно только прочитать

• System (системный) - файл с таким установленным атрибутом считается системным — таким, существование которого в неизменённом виде критически важно для нормальной работы системы.

• Hidden (скрытый) – этот атрибут позволяет скрыть файл, то есть Проводник не будет отображать его, если не включен специальный режим.

• Archive (архивный) - когда этот атрибут установлен, это означает, что файл был изменён со времени проведения последнего резервного копирования. ПО, с помощью которого выполняется резервное копирование, также отвечает за снятие этого атрибута.

2.6. Описание допустимых действий при работе с файлами.

• Открытие файла - самая частая операция с файлами и папками, в результате которой мы можем увидеть их содержимое.

• Операция переименования файлов.

• Операция перемещения файлов.

• Операции копирования файлов.

• Операция удаления файлов.

• Групповые операции над файлами.

• Операция создания файлов.

2.7. Описание инструментов, реализованных в исследуемом языке программирования, предназначенных для работы с файлами.

Открытие файла осуществляется с помощью функции fopen(), которая возвращает указатель на структуру типа FILE, который можно использовать для последующих операций с файлом.

```
1  FILE *f = fopen(name, type);
2  //some operations...
3  fclose(f);
4
```

name – имя файла, type — указатель на способ доступа к файлу:
r, w, a, r+, w+, a+.

Коды доступа задают права пользователей на доступ к файлу. Это число задает битовую шкалу из 9и бит, соответствующих строке биты: 876 543 210 (rwx rwx rwx). r - можно читать файл, w - можно записывать в файл, x -

можно выполнять программу из этого файла. 1 группа - эта права владельца файла, 2 - членов его группы, 3 - всех прочих.

Рассмотрим функции для чтения/записи файлов в ЯПВУ СИ:

Функция **fscanf** осуществляет ввод информации по формату:

```
fscanf(FILE *stream, const char *format, ...)

```

Функция **fprintf** осуществляет вывод информации по формату:

```
fprintf(FILE *stream, const char *format, ...)

```

2.8. Описание общего алгоритма подключения файла к пользовательской программе в терминах языка Pascal.

```
var
  SFileName, S: string;
  FText: Text;

begin
Write('File name: ');
ReadLn(SFileName);
Assign(FText, SFileName);
Reset(FText);
while not EOF(FText) do
  begin
    ReadLn(FText, S);
    WriteLn(S)
  end; { while not EOF(FText) }

Close(FText)
end.

```

Описание файловых переменных:

```
var
  f1: file of char; {typed file}
  f2: file of integer; {typed file}
  f3: file; {untyped file}
  f: text; {text file}

```

Для связи файла в коде программы и действительного файла на внешнем носителе используется процедура ASSIGN:

```
assign(myfile,'c:\text.txt');

```

Открытие файла

```
var f: text;

```

Рассмотрим процедуры, необходимые для работы с текстовым файлом в Паскале:

• процедура открытия существующего файла для чтения при последовательном доступе:

```
Reset (f);

```

• процедура добавления в конец:

```
Append (f);

```

• чтение из файла:

```
Read (f, list of variables);

```

• возврат в начало файла:

```
close (f);
reset (f); {start from the beginning}

```

• запись в текстовый файл:

```
Write (f, list of variables);

```

Закрытие:

```
Close (f);  {closing our file}

```

3. ПРАКТИЧЕСКАЯ ЧАСТЬ

Реализация общего алгоритма подключения файла к пользовательской программе для СИ.

В данной программе мы подключаем файл и считываем его содержимое посимвольно:

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>

int main(int argc, char* argv[])
{
  char ch;
  int fd;
  if((fd = open(argv[1], O_RDONLY)) == -1)
  {
    perror("Open");
  }
  while(read(fd, &ch, 1) == 1)
  {
    printf("%c", ch);
  }
  close(fd);
  return 0;
}

```

Демонстрация различных особенностей, характерных для работы с файлами в СИ.

В рассматриваемом языке программирования представлены два возможных уровня ввода-вывода (то есть одного из двух уровней управления доступом к файлам). Низкоуровневый ввод-вывод использует базовые функции ввода-вывода, предоставляемые системой. Стандартный высокоуровневый ввод-вывод предполагает применение стандартного пакета библиотечных функций С и определений из заголовочного файла «stdio.h». Стандарт ANSI С поддерживает только стандартный пакет ввода-вывода, поскольку в данном случае нельзя гарантировать, что все операционные системы могут быть

представлены одной и той же низкоуровневой моделью ввода-вывода. Таким образом аналогом функции fscanf/fprintf будут функции read/write.

Нижний уровень ввода-вывода (read и write)

```
int n_read = read(int fd, char *buf, int n);
int n_written = write(int fd, char *buf, int n);

```

Открытие файла:

```
int fd = open(char file_name[], int how_to_open);
//some operations...
close(fd);

```

Параметры как открыть: O_RDONLY, O_WRONLY, O_RDWR, O_APPEND(O_WRONLY|O_APPEND, O_RDWR|O_APPEND).

Если файл еще не существовал, то его нельзя открыть: open вернет значение (-1), сигнализирующее об ошибке. В этом случае его надо создать:

```
int fd = creat(char file_name[], int access_codes);
//some operations...
close(fd);

```

4. ЗАКЛЮЧЕНИЕ

Файловые системы Linux и Windows, как и сами системы в целом, действительно сильно отличаются. В Windows более дружелюбный интерфейс, адаптированный под базового пользователя, не разбирающегося в том как работает тот или иной алгоритм в ПО. Главные преимущества Linux: безопасность, бесплатность ПО, открытый код, популярность, количество программного обеспечения.

В заключение, хочу сказать, что благодаря данной курсовой работе я сумел почерпнуть интересный и необходимый для дальнейшего изучения языков программирования выского уровня багаж знаний. Мною были изучены несколько методических пособий по языкам программирования и их разработке, и это было очень увлекательно и занимательно. В дальнейшем планирую наращивать темпа изучения все новой и новой информации в данном русле.

5. СПИСОК ЛИТЕРАТУРЫ

1. Образовательный портал Московского Государственного Технического Университета имени Н.Э. Баумана bmstu.wiki;

2. Таненбаум Э. Современные операционные системы. 3-е изд. — СПб.: Питер, 2010. — 1120 с.

3. Керниган Б, Ритчи Д. Язык программирования Си. Издание 3-е, испр., 2003. — 253 с.

4. Орлов С.А. Теория и практика языков программирования: Учебник для вузов. Стандарт 3-го поколения. – СПб.: Питер, 2013. – 688 с.