Титульный лист методических рекомендаций
и указаний; методических рекомендаций;
методических указаний

Форма
Ф СО ПГУ 7.18.3/40

Министерство образования и науки Республики Казахстан

Павлодарский государственный университет имени С.Торайгырова

Кафедра информатики и информационных систем

# МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ И УКАЗАНИЯ

к выполнению лабораторных работ
по дисциплине Операционные системы
для студентов специальности 050602 – Информатика

Павлодар

Титульный лист методических рекомендаций и указаний; методических рекомендаций; методических указаний

Форма
Ф СО ПГУ 7.18.3/40

**УТВЕРЖДАЮ**
Декан ФФМиИТ
_____ _____
(подпись) (Ф.И.О.)
«___»_____20__г.

Составитель: к.п.н., доцент ПМУ М _

Кафедра Информатики и информа

# Методические рекомендации и указания

к выполнению лабораторных работ

по дисциплине Операционные системы
для студентов специальности 050602 – Информатика

Рекомендовано на заседании кафедры
«___»_____20____г., протокол №__

Заведующий кафедрой _____ Асаинова А.Ж. «____» _____20__г.
(подпись) (Ф.И.О.)
Одобрено УМС ФФМиИТ
«___»_____20__г., протокол №____
Председатель УМС _____ Муканова Ж.Г. «____» _____20__г.
(подпись) (Ф.И.О.)

## Тема3 Основные функции операционной системы

## Лабораторная работа № 1

Команды операционной системы MS DOS

**Цель:** Изучить основные команды MS DOS, отработать практические работы, применить основные команды на практике

### 1. Основные команды MS DOS

Создание текстовых файлов. copy con имя-файла

После ввода этой команды нужно будет поочередно вводить строки файла. В конце каждой строки нужно нажимать клавишу Enter, а после ввода последней – нажать клавишу F6 и затем Enter.

Удаление файлов. del имя-файла

Переименование файлов. ren имя-файла1 имя-файла2

Копирование файлов. copy имя-файла1 имя-файла2 или copy имя-файла1 [имя-каталога]

Соединение (конкатенация) файлов. copy имя-файла [+ имя-файла]. имя-файла

Команда смены текущего дисковода.

A: — переход на дисковод A:

B: — переход на дисковод B:

C:— переход на дисковод C:

Изменение текущего каталога. cd [дисковод:] путь

Просмотр каталога. dir [дисковод:][путь\][имя-файла] [параметры]

Создание каталога. md [дисковод:] путь

Уничтожение каталога. rd [дисковод:] путь

Удаление каталога со всем содержимым. deltree имя-файла-или-каталога [/Y]

Вывод файла на экран. type имя-файла

Очистка экрана монитора. cls

Вывод информации о дате и установка даты в компьютере. date

Вывод информации о времени и установка времени в компьютере. time [часы:минуты]

Получение информации о версии DOS. ver

**2. Задание к лабораторной работе**

Используя команды MS DOS, необходимо:

1. перейти в корневой каталог диска c
2. просмотреть содержимое диска c:.
3. перейти на диск d
4. создать в своем каталоге student на диске d: подкаталог [название группы]
5. создать в своем каталоге «название группы» на диске d: подкаталог [своё имя]
6. перейти в этот каталог
7. создать текстовый файл my_text1.txt. с произвольным текстом в 4…5 строк.
8. скопировать файл my_text1.txt с именем my_text2.txt.
9. просмотреть содержимое файла my_text1.txt.
10. перейти на диск d: в подкаталог student
11. скопировать в подкаталог [своё имя] все файлы из STUDENT с расширением *.txt. или *.doc
12. создать в каталоге «название группы» на диске d: подкаталог [new]
13. скопировать в подкаталог [new] все файлы из [своё имя]
14. переименовать папку [своё имя] в папку [very_new]
15. удалить подкаталог [new]
16. просмотреть содержимое каталога [very_new]
17. очистить экран.
18. вывести на экран номер версии dos.
19. вывести на экран текущую дату.
20. вывести на экран текущее время.

## Лабораторная работа №2.

Параметры команд и основных программ MS-DOS

**Цель:** Изучить применение параметров основных команд и программ.

1. **Параметры команд MS-DOS**

Спецификация команды: **DIR [н:][имя_файла[.тип]][/P][/W]**

Команда выдаёт информацию об элементах каталога (файлах и подкаталогах) - их имена, расширения, длину в байтах (для файлов), признак подкаталога <DIR> (для подкаталогов), время и дату создания, а также метку диска и объём свободного пространства на нём в байтах.

Параметр /P задаёт "постраничную" выдачу каталога размером с экран.

Часто бывает, что экрана "не хватает" для отображения всех строчек каталога. Чтобы иметь возможность внимательно просмотреть все строки, можно дать в конце текста команды специальный указатель /W. Параметр /W служит для компактной выдачи каталога.

Для каждой команды или программы MS-DOS может применятся параметр /?, который позволит посмотреть синтаксис команды и возможные параметры, а также параметры запуска программы MS-DOS.

2. **Программы и команды MS-DOS.**

   Edit – текстовый редактор.

   Fdisk – редактор разделов

   Scandisk, Chkdsk – утилита проверки диска

   Format – форматирование разделов.

3. **Задание к лабораторной работе**

1. Откройте эмулятор MS-DOS, перейдите в каталог вашей группы.
2. Просмотрите возможные **параметры** для каждой из изученных Вами команд и программ, запишите в тетрадь основные параметры.
3. Отработайте использование каждого из найденных параметров для всех известных Вам команд MS-DOS. Запишите их назначение.
4. Отработайте использование Scandisk/Chkdsk и Edit с заданием различных параметров.
5. Создайте загрузочную дискету MS-DOS.

4. **Контрольные вопросы.**

   1. В чём состоят основные отличия ОС MS-DOS и Windows?
   2. Название и структура файловой системы используемой MS-DOS?
   3. Какие типы (расширения) файлов, используемые MS-DOS вы знаете?
   4. Какие существуют ограничения на создание имен файлов ОС MS-DOS?

## Тема4 Процессы и потоки

## Лабораторная работа № 3.

Пакетные файлы в ОС DOC

**Цель:** научится использовать средства автоматизации часто повторяющихся действий путем создания BAT-файлов.

## Теоретические сведения

Командные файлы (BATCH-файлы, файлы пакетной обработки)- средство MS-DOS, позволяющее автоматизировать часто выполняемые действия пользователя. Командные файлы могут выполнять довольно сложную последовательность действий. Основой командных файлов служат команды MS-DOS. Такие файлы имеют расширение BAT и относятся к числу выполняемых файлов. Для выполнения такого файла необходимо ввести имя файла (можно без расширения) с необходимыми параметрами и нажать клавишу ENTER.

Командные файлы целесообразно применять в случае необходимости использования часто повторяющихся действий. Они являются наиболее удачным решением для реализации простых алгоритмов, связанных с операциями над файлами. Однако для реализации сложных алгоритмов язык командных файлов может оказаться недостаточным.

Командный файл состоит из последовательности строк, в каждой из которых может находиться либо вызов программы, либо вспомогательные команды. Создать файл можно в любом текстовом редакторе, набрав команды, а затем изменить расширение файла с .txt на.bat. В дальнейшем необходимому редактированию подвергается файл с расширением .bat в режиме редактирования оболочки NC (клавиша F3).

Большинство строк командного файла обрабатываются DOS так же, как если бы они вводились пользователем в командной строке. Вспомогательные команды служат для управления ходом работы командного файла.

Наиболее распространенные вспомогательные команды:

CLS. Очищает экран.

ECHO

Управляет выводом сообщений на экран в процессе выполнения файла. При формате вызова ECHO OFF отменяет вывод строк командного файла на экран при выполнении. Команда ECHO ON возобновляет их вывод на экран. Большинство командных файлов начинаются со строки @ECHO OFF. Символ "@" служит для отмены вывода на экран строки, следующей непосредственно за ним.

В остальных случаях команда ECHO выводит на экран текст, следующий за ней в строке. Команда ECHO text в режиме ECHO OFF выводит на экран сообщение text.

GOTO

Применяется для безусловного перехода к определенной строке командного файла. При выполнении команды GOTO:LABEL происходит переход к строке, начинающейся с текста :LABEL. LABEL – метка перехода – имя (последовательность до 8 символов) последовательности команд, которой команда GOTO передает управление.

Например:

```
Goto m1
   …..
m1:  cls
     echo Файл выполнен
```

IF

Служит для проверки условия во время выполнения командного файла.

Формат команды: IF УСЛОВИЕ КОМАНДА .

Команда будет выполнена в том случае, если условие будет истинно. Однако, допустим формат IF NOT УСЛОВИЕ КОМАНДА. При этом команда выполнится, если условие ложно. В качестве команды часто используется GOTO.

В качестве условия могут быть использованы выражения:

EXIST ФАЙЛ - истинно в том случае, если указанный файл существует.

СТРОКА1 = = СТРОКА2 - истинно, если строки совпадают. Обычно в одну из строк входят параметры командного файла. Например, при условии if not string1= = string2 dir команда dir выполнится, если две строки не совпадают хотя бы в одном символе.

ERRORLEVEL число – команда выполняется в том случае, если код завершения предыдущей команды или программы больше или равен указанному числу.

PAUSE

Приостанавливает выполнение командного файла до нажатия любой клавиши. Если нажать клавиши Ctrl+Break или Ctrl+C, выполнение командного файла будет прервано.

REM

Строки, начинающиеся с REM, при выполнении командного файла игнорируются. В них можно записывать комментарии.

FOR

Многократное (циклическое) выполнение заданной команды для совокупности файлов.

Формат команды: FOR %%x IN (список) DO команда %%x

При выполнении КОМАНДА будет выполнена для каждого файла из СПИСКА. Эту команду можно использовать, если программа, которую нужно вызвать, не поддерживает "джокеров" - символов "*" и "?" в именах файлов.

Пример:

```
Cd\ text
FOR %%1 IN (TEXY?.TXT) DO TYPE %%1
```

Такая последовательность команд выдаст содержимое всех файлов каталога text, удовлетворяющих шаблону TEXY?.TXT с помощью команды TYPE.

Эту команду можно использовать и в командной строке, но при этом вместо %%x нужно писать %x.

ПАРАМЕТРЫ КОМАНДНОГО ФАЙЛА - слова, идущие через пробел при вызове командного файла в командной строке вслед за именем самого файла.

Пример: compare.bat example.txt example2.txt

При запуске командного файла фактические параметры указываются с помощью выражений %x, где x - цифра от 1 до 9. %0 означает имя самого командного файла compare.bat, %1 – example.txt и т.д. Так, если в приведенном выше примере файл compare.bat содержит строку type %1 , то во время выполнения она заменится на type example.txt.

ASK

Команда выводит указанное сообщение и ждет, пока пользователь не введет один из указанных в списке символов. Значение переменной ErrorLevel устанавливается равным номеру введенного символа в списке.

Пример:

```
@echo off
cls
echo View files
echo 1 - 1.txt
echo 2 - 2.txt
ask "Code -", 12
if errorlevel 2 type 2.txt
if errorlevel 1 type 1.txt
pause
```

результатом выполнения файла будет выведение на экран содержимого файла 1.txt или 2.txt, в зависимости от ответа, полученного на запрос.

SHIFT

Сдвиг параметров командного файла. После выполнения этой команды все значения формальных параметров перемещаются на одну позицию, т.е. %1 примет значение %2, %2 - %3 и т.д.

Следует заметить, что при сравнении строк, содержащих символы параметров 0% - %9, следует проявлять осторожность. Например, сравнение второго параметра со строкой CCC не следует делать так: IF %2 = = CCC. Дело в том, что если в командном файле меньше двух параметров, то %2 будет замещено пустой строкой, и при выполнении команды возникнет ошибка. Правильнее сравнение выглядит так: IF -%2 = = -CCC

Некоторые команды MS-DOS, необходимые в написании командных файлов:

Смена текущего каталога осуществляется командой CD имя_каталога

Создание каталога осуществляется командой MD имя_каталога.

Копирование файла - командой COPY имя_файла1 имя_файла2.

Удаление файла - командой DEL имя_файла.

Удаление каталога – командой DELTREE имя_каталога

Файлы объединяются командой COPY файл1 + файл2 файл3, после выполнения которой файл3 содержит данные, находившиеся в файлах файл1 и файл2.

**Указания к содержанию отчета по лабораторной работе №3**

Задания для работы выбираются в соответствии с вариантом.

Состав выполненной работы:

1.рабочее задание;

2.практическая часть:

а) текст программы;

б) протокол работы программы.

## Пример отчета по лабораторной работе №3.

Создать командный файл с именем BATCH.BAT, выполняющий действия с файлом, имя которого передано как параметр в строке вызова:

1.Отключение режима отображения на экране выполняемой команды

2.Вывод на экран “ Поиск файла по заданному имени ”

3.Вывод сообщения " Укажите имя файла ", если файл BATCH.BAT был вызван без параметров.

4.Вывод сообщения " Файла с таким именем не существует ", если указанный файл не найден.

5.Вывод содержимого указанного файла на экран.

*Текст программы*

```
cls
@echo off
echo Поиск файла по заданному имени
if -%1 == - goto no_p
if not exist %1 goto no_ex
echo Текст файла %1
type %1
goto exit
:no_p
echo Укажите имя файла
goto exit
:no_ex
echo Файла с именем %1 не существует
:exit
```

*Протокол работы программы*

а) Файл BATCH.BAT: вызов без параметров.

```
Поиск файла по заданному имени
Укажите имя файла
```

б) Файл BATCH.BAT: неправильно указано имя файла или файл не существует.

```
Поиск файла по заданному имени
Файла с именем 1.txt не существует
```

в) Файл BATCH.BAT: вывод содержимого указанного файла.

```
Поиск файла по заданному имени
Текст файла 1.txt
Мама мыла раму.
```

## Задание

0. Создать командный файл BATCH0.BAT без и с командой SHIFT, выполняющий действия:

1. Отключение режима отображения на экране выполняемой команды
2. Вывод на экран: "Копирование файлов по маске в корневой каталог"
3. Копирование файлов, имеющих маску TEXT?.DOC из каталога TEXT в корневой: а) с помощью команды SHIFT, б) без команды SHIFT.
4. Вывод на экран: "Файл (имя файла) скопирован"
5. Удалить из каталога TEXT скопированные файлы.

1. Создать командный файл BATCH1.BAT, выполняющий действия:

1. Отключение режима отображения на экране выполняемой команды
2. Вывод на экран: "Копирование и удаление файла"
3. Создание на диске C: каталога DIR1, и в нем создание каталога DIR2
4. Копирование файла с именем TEXT1.TXT из каталога C:\ALPHA\BETTA\GAMMA в файл с именем TEXTNEW.TXT в каталоге C:\DIR1\DIR2
5. Удаление исходного файла
6. Вывод на экран: “Файл скопирован и удален”
7. Пауза до нажатия клавиши

2.Создать командный файл с именем SUMMA.BAT, выполняющий действия:
1. Отключение режима отображения на экране выполняемой команды
2. Вывод на экран “Объединение и переименование файлов
3. Объединение содержимого файлов ANEW.PAS и BNEW.PAS, находящихся в каталоге C:\D1, в файл CNEW.PAS в каталоге C:\D2
4. Вывод содержимого файла CNEW.PAS на экран
5. Ожидание нажатия клавиши
6. Переименование файлов ANEW.PAS и BNEW.PAS в AOLD.PAS и BOLD.PAS соответственно.
7. Вывод на экран: "Задание выполнено"

3.Создать командный файл с именем _EXIST.BAT, выполняющий действия:
1. Отключение режима отображения на экране выполняемой команды
2. Вывод на экран “Копирование файла в случае его отсутствия на дискете”
3. В случае отсутствия файла SIMP.FOR на диске A: (или каталоге A) скопировать его туда из каталога C:\FOR и вывести на экран: "Файл simp.for скопирован на диск A: (в каталог A)"
4. Если файл SIMP.FOR уже существует на дискете или в каталоге, вывести на экран: "Файл simp.for уже существует на дискете (в каталоге)"

4. Создать командный файл с именем BATCH4.BAT, выполняющий различные действия в зависимости от переданного параметра в строке вызова:
1. Отключение режима отображения на экране выполняемой команды
2. Вывод на экран “Создание, просмотр содержимого и удаление каталогов”
3. Создание каталога C:\MYDIR и копирование в него всех .com и .exe файлов с диска C:
4. Создание в каталоге C:\MYDIR каталога \NEWDIR и копирование в него всех .com файлов с диска C:
5. Вывод на экран содержимого каталога C:\MYDIR
6. Удаление каталога C:\MYDIR\NEWDIR
7. Удаление каталога C:\MYDIR

5. Создать командный файл с именем BATCH5.BAT, выполняющий следующие действия:
1. Отключение режима отображения на экране выполняемой команды
2. Вывод на экран “Просмотр информации о студентах”
3. В каталоге C:\MYDIR создает подкаталог TEXT.
4. В созданном каталоге создать файл TEXT1.DOC, содержащий следующую информацию: ФИО, Павлодар, ПаУ, группа ЗАСУ-31.
5. Выдает сообщение "Информация о студентах:" и выдает на экран содержимое файла TEXT1.DOC.

6. Создать командный файл с именем BATCH6.BAT, выполняющий следующие действия:
1. Отключение режима отображения на экране выполняемой команды
2. Вывод на экран “ Копирование файлов с расширением BAT ”
3. Создание каталога C:\MYDIR и копирование в него всех .bat - файлов с диска C:
4. Вывод на экран содержимого каталога C:\MYDIR
5. Удаление каталога C:\MYDIR

7. Создать командный файл с именем BATCH7.BAT, выполняющий следующие действия:
1. Отключение режима отображения на экране выполняемой команды
2. Вывод на экран “ Перенос файлов с расширением BAT ”
3. Создание каталога C:\MYDIR и копирование в него всех .bat - файлов из каталога C:\student\test.
4. Удаление всех .bat - файлов из каталога C:\student\test.
5. Вывод на экран содержимого каталога C:\MYDIR

6. Удаление каталога C:\MYDIR

8. Создать командный файл с именем BATCH8.BAT, выводящий на экран содержимое файла TEXT1.DOC до тех пор, пока не будут нажаты клавиши Ctrl-C.

9. Создать командный файл с именем BATCH9.BAT, выводящий на экран день недели (1-понедельник, 2 – вторник и т.д.) в зависимости от полученного ответа на выдаваемый запрос с помощью а) команды ECHO, б) из текстового файла.

## Тема6 Файловые системы

## Лабораторная работа № 4.

Файловая система

**Цель:** научится использовать функции типа дескриптора в работе с файловой системой

### Теоретические сведения

УПРАВЛЕНИЕ ФАЙЛАМИ В ОС

ОС предполагают по крайней мере два различных системных вызова для всех операций с файлами и записями.

1. Набор функций для управления файлами и записями, совместимый с CPM и называемый функциями типа FCB.
2. Функции управления файлами и записями, обеспечивающие совместимость с UNIX и называемые функциями типа дескриптора (handle-функции). Эти функции позволяют выполнять операции с файлами путем передачи ОС 16-бит маркера – *дескриптора (handle)*.

Обращение к файлам происходит с помощью ASCII-Z строк. ASCII-Z строка представляет собой обычную строку в ASCII формате, в которой указывается путь к определенному файлу. В конце ASCII-Z строки всегда стоит 0, что сигнализирует системе, что эту стоку надо использовать как путь.

В случае успешного выполнения какой-либо операции с файлами (открытие, чтение, копирование и т.д.) ОС возвращает через регистр AX дескриптор. Для дальнейшей работы с данным файлом достаточно лишь передать системе его дескриптор через регистр BX

Если файловая операция выполняется успешно, то флаг переноса CF сброшен. В обратном случае AX содержит код ошибки, описывающий причину возникновения ошибки.

### Задание

1. Создать файл HANDLE-способом. Записать в файл 20 случайных чисел диапазона 33 .. 77
2. Создать 10 новых файлов HANDLE-способом. После создания удалить 5 из них случайным образом
3. Создать файл HANDLE-способом. Скопировать в него содержимое произвольного текстового файла
4. Создать файл HANDLE-способом. Вывести на экран дату и время создания данного файла
5. Создать файл HANDLE-способом. Установить данному файлу произвольную дату и время создания
6. Создать файл HANDLE-способом. Записать в него три произвольные текстовые строки. Вывести на экран размер полученного файла
7. Найти в желаемом каталоге все файлы типа «*.pas». Записать результат поиска в файл HANDLE -способом.
8. Искать в желаемом каталоге все файлы типа «*.tmp» и заменять их имена на «xx.$$$», где xx – 01, 02, 03 и т.д.
9. Скопировать содержимое всего каталога в произвольный каталог
0. Найти в желаемом каталоге все файлы типа «*.txt». Вывести результат поиска на экран в отсортированном виде

## Пример использования функций файловой системы

```
function OpenFile (nFileName: string): word;{ Функция открытия файла Handle-способом }
var p: pointer; k: word;
begin
nFileName:=nFileName+#0; p:=@nFileName;
asm
push ds; push dx;  mov ah,3dh; mov al,0;  lds dx, p; add dx,1;
int 21h;
jc @@@1; jmp @@@2;
@@@1: mov ax,0;
@@@2: mov k,ax;
pop dx; pop ds;
end; OpenFile:=k;
end;
function CreateNewFile (nFileName: string): word; { Функция создания нового файла Handle-
способом }
var p: pointer; k: word;
begin
nFileName:=nFileName+#0; p:=@nFileName;
asm
push ds; push dx;  mov ah,3ch; mov cx,0;  lds dx, p; add dx,1;
int 21h;
jc @@@1; jmp @@@2;
@@@1: mov ax,0;
@@@2: mov k,ax;
pop dx; pop ds;
end; CreateNewFile:=k;
end;
procedure CloseFile (handle: word); assembler;{ Процедура закрытия файла Handle-способом }
asm
mov ah,3eh; mov bx,handle; int 21h;
end;

function OutDataInFile (handle: word; xData: string): boolean; { Функция записи текстовой строки в
                                                                  файл Handle-способом }
var p: pointer; k,len: word;
begin
p:=@xData; len:=length(xData);
asm
push ds; push dx; mov bx,handle; mov ah,40h; mov cx,len; lds dx, p; add dx,1;
int 21h;
jc @@@1; jmp @@@2;
@@@1: mov ax,0;
@@@2: mov k,ax;
pop dx; pop ds;
end;
if k=0 then OutDataInFile:=false  else OutDataInFile:=true;
end;
function GetCharFromFile (handle: word; var xData: char): boolean;{ Функция чтения символа из
                                                                    файла Handle-способом }
var p: pointer; k: word;
begin
p:=@xData;
asm
push ds; push dx; mov bx,handle; mov ah,3fh; mov cx,1; lds dx, p;
int 21h;
jc @@@1; jmp @@@2;
```

```
@@@1: mov ax,0;
@@@2: mov k,ax;
pop dx; pop ds;
end;
if k=0 then GetCharFromFile:=false else GetCharFromFile:=true;
end;

{ Основная часть программы }
var k: word; c: char;
begin
k:=CreateNewFile (‘1.txt’);
{  k:=OpenFile (‘c:\1.ttt’); }
if OutDataInFile (k,’Привет, Малышка!’) then writeln(‘Ok!’);
{  if GetCharFromFile (k,c) then writeln©; }
if k<>0 then CloseFile (k);
readln;
end.
```

## **Лабораторная работа №5.**

Операционная система LINUX

**Тема:** Основы работы с OS LINUX
**Цель работы:** Ознакомление с основными командами Linux, получение навыков по их использованию.

### Общие сведения о Linux

## Что такое Linux?

Итак, что же это все-таки такое - Linux? Linux создавался как операционная система для IBM совместимых компьютеров (Теперь Linux перенесен на многие другие платформы, в том числе на 680x0, Альфу и многопроцессорные рабочие станции). Linux создан и продолжает создаваться благодаря усилиям программистов, разбросанных вокруг всего мира. Целью его разработки было создать операционную систему, принадлежащую к клону UNIX, но свободную от каких-либо коммерческих авторских прав, которую могли бы использовать программисты всего мира. Вообще-то Linux стартовал как хобби одного человека - финского студента Линуса Торвальда. Тогда его целью было создание замены MINIX. Это была похожая на UNIX учебная операционная система для компьютеров с процессорами фирмы Intel.

## Что такое UNIX?

Ну, хорошо, теперь мы знаем, что такое Linux. А что такое Unix? Unix это тоже операционная система. Но эта операционная система не привязана к определенному типу компьютера. Один из самых интересных фактов из биографии UNIX это то, что ее первый прототип был написан в 1969 для машины DEC PDP-7 на ассемблере. В 1973 году она была переписана на Си. Благодаря этому она получила легкую переносимость на другие типы машин.

В те времена UNIX распространялся не очень быстро - эта ОС требовала больших ресурсов. Но благодаря легкости переноса на другие машины и удачной концепции UNIX распространялся все шире. Суть концепции UNIX в том, что задачи решаются не благодаря большим и мощным программам, а благодаря взаимодействию небольших программ. Сегодня практически на всех машинах, которые используются для научных или других применений, где требуется многозадачность и многопользовательский режим стоят

или UNIX или UNIX подобные ОС. Широко разрекламированная Windows NT это тоже UNIX подобная система в исполнении фирмы Microsoft.

## Чем Linux отличается от UNIX?

Главные отличия LINUX-а от UNIX-а:

- **Цена.** Коммерческие UNIX системы стоят 1000 - 3000 USD. Linux распространяется бесплатно или для коммерческих дистрибутивов по сравнительно низкой цене.
- **Лицензионная политика.** Linux распространяется вместе с исходными текстами и под лицензией, которая не разрешает использовать Linux не распространяя исходных текстов. Эта политика постоянно поддерживает цену коммерческих дистрибутивов Linux на низком уровне. Эта политика делает также невозможным использования тактики имени Microsoft - использование недокументированных возможностей системы.
- **Портативность** Linux с самого начала был предназначен для работы на IBM совместимых компьютерах. Отсюда его невысокие требования к ресурсам.

## Краткое описание команд Linux

*Загрузка операционной системы*

Данная работа выполняется с использованием ОС LINUX, работающей на виртуальной машине управляемой программой Microsoft Virtual PC в операционной среде Windows 2000. Запуск операционной системы Linux производится в следующем порядке.

1. Запускается оболочка MS Virtual PC как задача Windows. Ярлык для запуска может находиться или на рабочем столе Windows, или в меню «Пуск»
2. Запускается виртуальная машина с предустановленной на ней ОС Linux.
3. После полного запуска ОС выполняется регистрация в системе.
4. Устанавливается IP адрес для вашей виртуальной машины (если он еще не установлен). Для это изменяется одна из строк в файла /ETC/SysConfig/NetWork-Scripts/ifcfg-eth0. Адрес для вашей машины должен быть следующий: 10.10.146.190+xx, где xx – порядковый номер компьютера.

*Вход в систему*

Теперь надо ввести имя пользователя и пароль. По имени пользователя система опознает вас как одного из пользователей, которые могут работать в системе одновременно или поочередно. Для каждого пользователя определяется каталог по умолчанию, именуемый *рабочим, или домашним, каталогом (home directory).* Многие пользователи имеют доступ к ограниченному числу каталогов и команд — главным образом для того, чтобы они не могли заглядывать в файлы друг друга.

Для данной лабораторной работы следует использовать имя пользователя – **root,** а пароль – **rootuser,** либо не использовать пароль, а в качестве логина использоать своё имя.

*Внимание!* При вводе имени и пароля следует учитывать регистр букв. После регистрации следует обязательно сменить пароль, это делается командой «passwd roor».

*Ввод команд*

Ввод команд в Linux выглядит примерно так же, как в DOS и других операционных системах, ориентированных на ввод в командной строке. Linux, как и UNIX, чувствительна к регистру, поэтому если система не воспринимает какую-либо команду, проверьте, в правильном ли регистре вы ввели ее. Как правило, команда выполняется после нажатия клавиши <Enter>.

## Вызов истории команд

В Linux есть средство повторного обращения к уже выполненным командам, которое не прерывается даже при выключении компьютера. Предыдущая команда вызывается после нажатия клавиши <Up>, а для ее выполнения надо нажать <Enter>. Для вывода всего списка примененных команд воспользуйтесь командой history:

**[tackett@web~] $ history**
**1 clear**
**2 adduser**
**3 history**

Чтобы выполнить команду из хронологического списка, вызывайте с помощью клавиши <Up> предыдущую команду до тех пор, пока в командной строке не появится нужная, или же нажмите <!> и введите номер нужной команды. Например, чтобы повторно выполнить команду adduser из представленного выше списка, введите

**[linux_lab@ais] $ !2**

Максимальное число команд в хронологическом списке задается в пользовательском конфигурационном файле .profile.

## Основные команды Linux

**Команда справки *man***

Для получения справки по той или иной команде Linux воспользуйтесь командой man. В ответ Linux открывает на нескольких, сменяющих друг друга экранах описание нужной команды. Если не помните точно имени нужной команды, введите команду man с параметром -k, затем ключевое слово для поиска нужной команды. Система выполнит поиск в своих файлах справки, содержащей это ключевое слово. Для этой команды имеется также псевдоним apropos.

Например, если ввести команду man ls, Linux выведет на экран справку о команде Is, в том числе обо всех ее параметрах. По команде man -k cls выводится список всех команд, в справке, о которых есть слово cls. Команда apropos cls аналогична команде man -k cls.

***Команды для работы с каталогами***

В Linux есть много команд для работы с каталогами. Как и в других операционных системах, в которых вам, возможно, приходилось работать, каталоги в Linux можно удалять, создавать, перемещать, а также выводить информацию об их состоянии.

**Смена текущего каталога с помощью команды *cd*.**

В Linux, как и в DOS, файлы хранятся в каталогах, организованных в древовидные структуры. Файл можно указывать в виде пути из корневого каталога, обозначаемого символом /, до файла. Таким образом, полное имя конфигурационного файла .emacs, принадлежащего пользователю jack, может иметь вид /home/ jack/.emacs.

Тем, кто привык работать с файлами DOS, длина имен которых не превышает восьми символов, а расширений — трех, приятно будет узнать, что в Linux подобных ограничений нет.

В Linux есть понятие рабочего каталога пользователя. Рабочий каталог обычно обозначается символом ~ (тильда). Например, команда копирования файла из текущего каталога в рабочий может иметь вид cp .emacs ~

Для перемещения по дереву каталогов Linux применяется команда cd. Для перехода в рабочий каталог эта команда вводится без параметров. Для перехода из одного каталога в другой формат команды тот же, что и DOS: cd new-directory, где new-directory — новый каталог, в который следует перейти. Кроме того, в Linux текущий каталог представляется одной точкой (.), каталог-родитель — двумя (. .) — и, конечно же, в этом DOS наследует UNIX и Linux, а не наоборот.

**Будьте внимательны с символом разделителя каталогов. В DOS для этого применяется обратная косая черта (\), которая в Linux служит указателем продолжения команды с новой строки. В Linux каталоги разделяются прямой косой чертой (/). Кроме того, в DOS не имеет значения, отделены ли параметры (.) и (. .) пробелами от имени команды, в то время как в Linux это важно, Linux не поймет команды cd. ., правильный формат которой — cd ... В Linux между командой и параметром обязательно должен быть пробел.**

**Вывод информации о файлах и каталогах с помощью команды *ls***

ls — сокращение от list (список). В Linux по этой команде на экран выводится список файлов. Это аналог команды dir из DOS (которую можно применять и в Linux) для вывода списка файлов в каталоге.

Чтобы указать, как именно выводить список файлов, каких файлов и с какой информацией о файлах, придется ввести команду ls с параметрами. Чаще всего применяется параметр -1a, по которому выводится полная информация о каждом файле каталога. По команде is -la выводится подробная информация о файлах текущего каталога. По команде Is emacs выводится только имя этого файла, по команде же Is -la emacs — полная информация о нем.

**Создание каталога с помощью команды *mkdir***

Поскольку структура каталогов составляет основу файловой системы, в Linux имеется также команда создания каталога mkdir. В отличие от DOS, где можно воспользоваться псевдонимом данной команды MD, в Linux надо вводить ее полное имя. В качестве параметра указывается имя создаваемого каталога, как в следующем примере: **mkdir backup**

**Удаление каталогов с помощью команды *rmdir***

Каталоги в Linux удаляются с помощью команды rmdir, в качестве параметра которой указывается удаляемый каталог. Linux может удалить только пустой каталог. Например, если в каталоге /backup есть два подкаталога, команда rmdir /backup выполнена не будет. Чтобы удалить один из подкаталогов — /jack, — сначала по команде rmdir /backup/jack/* из него удаляются все файлы, затем с помощью команды rmdir /backup/jack — он сам.

**ВНИМАНИЕ**

**С помощью команды rmdir нельзя удалить непустой каталог, но это можно сделать с помощью команды rm с параметром -r. Например, по команде rm -r * из текущего каталога будет удалено все, включая подкаталоги. Будьте внимательны, пользуясь этой командой, ибо, удалив каталог, нельзя будет восстановить ни его, ни содержавшиеся в нем файлы.**

***Команды работы с файлами***

В Linux нет принципиального различия между файлами и каталогами, поэтому для работы с теми и другими применяются одни и те же команды.

**Копирование файлов с помощью команды *cp***

Команда cp аналогична команде COPY из DOS. Она применяется для копирования одного или нескольких файлов из одного каталога в другой. Синтаксис команды:

**cp from-filename to-filename,**

где from-filename — исходный файл; to-filename — файл, в который происходит копирование.

Чтобы команда была выполнена, надо указать оба параметра. Чтобы скопировать файл с тем же именем в качестве второго параметра, ставится точка (.). В этом отличие от DOS, где второй параметр в подобном случае просто опускается.

По команде cp fredl fredl.old создается резервная копия файла fredl с именем fredl.old. По команде же cp ~fredl.old /backup/jack файл fredl. old копируется из рабочего каталога в каталог /backup/jack. Рабочий каталог представлен символом ~.

**Конкатенация файлов с помощью команды cat**

Команда cat считывает содержимое указанных файлов и выводит его на стандартный вывод. Если имя файла в командной строке не указано, то ожидается вывод ввод данных со стандартного ввода.

Синтаксис:

**cat [-u] [-s] file…**

Опции

s - не надо выводить сообщение об отсутствии аргументов;

u – вывод производится небуферизованный, т.е. символы из входного файла сразу поступают на стандартный вывод.

**Перемещение файлов с помощью команды *mv***

По команде mv, аналогичной команде MOVE из DOS, файлы перемещаются из одного каталога в другой. Действие этой команды аналогично действию команды копирования с последующим удалением исходных файлов. Команда mv не создает копий файлов.

Синтаксис команды mv:

**mv from-filename to-filename,**

где from-filename — исходный файл; to-filename — новый файл.

По команде mv fredl |redl.old создается резервная копия файла fredl с именем fredl.old, затем удаляется исходный файл |redl. По команде же mv -fredl.old /backup/jack файл fredl.old перемещается из рабочего ката-уга в каталог /backup/jack.

**Удаление файлов с помощью команды *rm***

Файлы в Linux удаляются по команде rm. Это опасная команда, потому что удаленный файл восстановить невозможно. Для безопасной работы следует пользоваться следующим форматом этой команды:

**rm -i filename,**

здесь filename — имя удаляемого файла; -i — параметр, указывающий на необходимость подтвердить удаление файла.

Например, по команде rm fredl файл fredl будет просто удален, по команде же rm -i il он будет удален только после подтверждения пользователем необходимости удаления.

**Вывод содержимого файла с помощью команды *more***

По команде more на экран выводится содержимое текстового файла, при этом нет необходимости запускать текстовый редактор, распечатывать файл или нажимать клавишу паузы во время вывода текста на экран. Например, для вывода на экран содержимого конфигурационного файла emacs вводится команда:

**more .emacs.**

Недостаток этой команды в том, что невозможно пролистать информацию в обратном направлении. Но, этот недостаток устранен в других командах, которые мы рассмотрим ниже.

**Команда /ess—усовершенствование more**

По команде less информация выводится в окно терминала. Имя этой команде дано в противоположность команде more, поскольку в команде less пролистывание текстового файла возможно в обоих направлениях (игра слов: more — больше, less — меньше.*)*. Синтаксис команды less:

**less файл**

**Команда поиска файлов *find***

Если вы не можете найти требуемый файл с помощью команды Is, воспользуйтесь командой find. Команда find исключительно полезная вещь, что делает ее одновременно одной из самых сложных в использовании команд. Использование команды find включает три этапа, которые в свою очередь могут состоять из одного или нескольких этапов.

- Где искать
- Что искать
- Что делать, когда файл найден

Если вы знаете имя файла, но не знаете, где он находится в структуре каталогов Linux, то самым простым способом использования команды find для поиска такого файла будет команда:

**find / -name *filename* -print**

Будьте осторожными при поиске от корня — в больших системах такой поиск может занять слишком много времени, так как будет просматриваться каждый каталог, каждый диск, включая подключенные сетевые диски.

Возможно, более приемлемым будет поиск по нескольким каталогам. Например, если вы знаете, что файл, вероятнее всего, находится в каталогах /usr или /usr2, воспользуйтесь следующей командой:

**find /usr /usr2 -name *filename* -print**

В команде find можно использовать множество различных параметров. Список параметров команды приведен в таблице.

**Таблица. Параметры команды find**

| Команда | Описание |
|---|---|
| **-name *file*** | Параметр file может быть именем или шаблоном, содержащим символы подстановки. Если это шаблон, то для обработки выбирается каждый файл, чье имя удовлетворяет этому шаблону |
| **-links *n*** | Для обработки выбираются все файлы, на каждый из которых имеется n или больше ссылок |
| **-size n [c]** | Для обработки выбираются все файлы, размер которых равен или больше n 512-байтных блоков. Если к размеру добавлен символ с, то выбираются файлы, которые состоит из n или больше символов |
| **-atime *n*** | Для обработки выбираются все файлы, к которым осуществлялся доступ за последние n-дней. Обратите внимание, что сама команда find осуществляет доступ к файлам, поэтому изменяет время последнего доступа к файлу |

| | |
|---|---|
| **-exec *cmd*** | Для каждого файла, удовлетворяющего критериям поиска, выполняется команда Linux, принимающая в качестве параметра имя найденного файла. Для использования команды -exec необходимо запомнить два простых правила: в команде имя найденного файла представляется {}, а команда должна заканчиваться последовательностью символов \;. Предположим, вы зарегистрировались как администратор и создали каталог, поэтому все файлы в этом каталоге принадлежат администратору. Чтобы сделать так, чтобы всеми файлами владел пользователь jack, необходимо выполнить команду:<br>find /home/jack -exec chown jack {} \; |
| **~print** | Эта наиболее часто используемая команда просто отображает имена всех найденных файлов |

---

Команда find позволяет выполнять множество логических операций. Например, если вы хотите выбрать все файлы, которые нельзя представить одним шаблоном, можно воспользоваться параметром *or* (-o):

**find /home ( -name *filel -o* -name *file2* ) –print**

## Задание

1. Вывести оглавление домашнего (рабочего) каталога, предназначенного для пользователей (**/web_server**).
2. Создать в текущем каталоге каталог с именем **name**, где **name** - ваша фамилия маленькими латинскими буквами.
3. Перейти в созданный каталог.
4. Создать каталог **MLKX** в текущем каталоге.
5. Скопировать в каталог **MLKX** файл **xyz.text** из каталога **/web_server/Metodika/laba1/**.
6. Вывести файл **xyz.text** на экран.
7. Создать каталог с именем **FFW**.
8. Скопировать файл **xyz.text** в каталог **FFW** с именем **xyz1.text.**
9. Перейти в каталог **FFW** и просмотреть содержимое файла **xyz1.text**.
10. Переименовать файл **xyz1.text** в файл с именем **xyz2.text**.
11. Объединить файлы **xyz.text** и **xyz2.text** в файл с именем **xyz3.text** в каталоге **FFW**.
12. Просмотреть файл **xyz3.text**.
13. *Продемонстрировать содержимое файла преподавателю*.
14. Произвести поиск файла **xyz.text** с помощью команды **find**.
15. Удалить файлы **xyz.text, xyz2.text и xyz3.text.**
16. Удалить каталог **FFW**.
17. Удалить каталог **MLKX**.
18. Удалить каталог **name**.

## Тема7 Управление вводом - выводом

## Лабораторная работа № 6.

Файловая система операционной системы LINUX

**Тема:** Основы работы с ФС ОС LINUX
**Цель работы:** Изучить основные свойства и возможности файловых систем ОС Linux.

**Ход выполнения работы:**

1. Изучение теоретического материала;
2. Применение изученных команд и параметров на практике;
3. Выполнение всех пунктов задания;
4. Оформление подробного отчёта.

**Требование к отчёту:**

1. Формат А4;
2. Оформление в **строгом** соответствии с СТП;
3. Наличие листа с заданием в начале отчёта;
4. Подробное описание всех выполняемых действии по пунктам;
5. Листинг файла report.txt в приложении с пояснениями;
6. Приложение отчета и файла report.txt на электронном носителе.

***1. Файлы и каталоги. Дерево каталогов***

В свое время, при использовании DOS вводилось определение файла как поименованной области данных на диске - на то DOS и дисковая операционная система. В Linux понятие файла значительно расширено. Практически все, с чем вы имеете дело в Linux, является файлом. Команды, которые вы вводите с клавиатуры, - это файлы, которые содержат программы. Устройства вашего компьютера - это тоже файлы. Грубо говоря, файл – это последовательность битов, а жесткий диск - просто смесь нулей и единиц.

Linux представляет биты так, как вам понятно, и в этом заключается одна из ее основных функций - управление файловой системой.

Файловая система - способ организации и представления битов на жестком диске. Большинство файловых систем Unix-подобных операционных систем сходны между собой. Файловая система Linux - **ext2 (ext3)** - очень похожа на файловую систему ufs.

К основным понятиям файловых систем в мире Unix относятся:

1. Блок загрузки (boot block).
2. Суперблок (superblock).
3. Индексный (информационный) узел (inode).
4. Блок данных (data block).
5. Блок каталога (directory block).
6. Косвенный блок (indirection block).

**Блок загрузки** содержит программу для первоначального запуска Unix. **В суперблоке** содержится общая информация о файловой системе. В суперблоке также содержится информация о количестве свободных блоков и информационных узлов. Для повышения устойчивости создается несколько копий суперблока, но при монтировании используется только одна. Если первая копия суперблока повреждена, то используется его резервная копия.

**В индексном (информационном) узле** хранится вся информация о файле, кроме его имени. Имя файла и его дескриптор (номер информационного узла - inode) хранятся в блоке каталога. В информационном узле есть место только для хранения нескольких блоков данных. Если же нужно обеспечить большее количество, то в этом случае динамически выделяется необходимое пространство для указателей на новые блоки данных. Такие блоки называются **косвенными.** Для того, чтобы найти блок данных, нужно найти его номер в косвенном блоке.

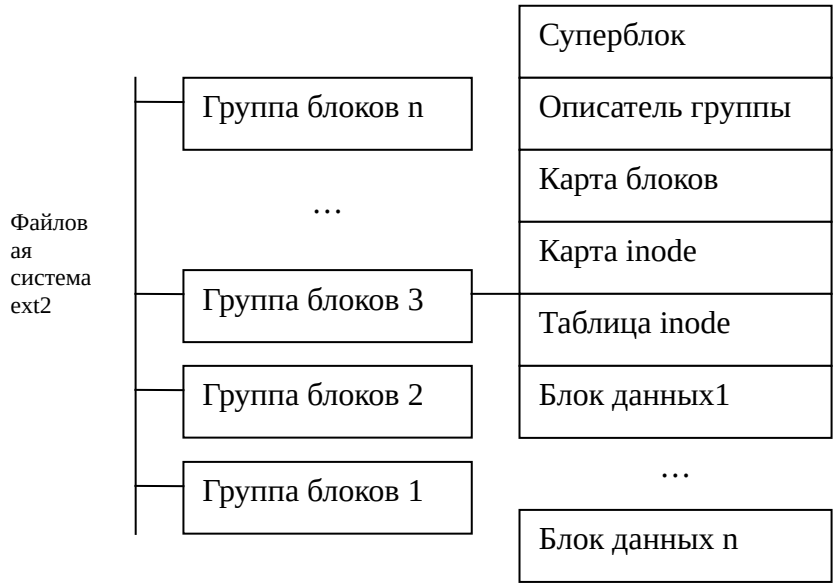


Рисунок 1 Структура файловой системы

Файловая система ext2 имеет следующую структуру (см. рис. 1):

1. Суперблок.
2. Описатель группы.
3. Карта блоков.
4. Карта информационных узлов.
5. Таблица информационных узлов.
6. Блоки данных.

Суперблок, как уже отмечалось выше, содержит информацию обо всей файловой системе. Он имеется в каждой группе блоков, но это всего лишь копия суперблока первой группы блоков: так достигается избыточность файловой системы.

Описатель (дескриптор) группы содержит информацию о группе блоков. Каждая группа имеет свой дескриптор группы. Карты блоков и информационных узлов - это массивы битов, которые указывают на блоки или информационные узлы соответственно. Таблица информационных узлов содержит информацию о выделенных для данной группы блоков в информационных узлах.

Блоки данных - это блоки, содержащие реальные данные.

Что касается файлов, то в операционной системе Linux существует четыре типа файлов:

1. Файлы устройств.

2. Каталоги.

3. Обычные файлы.

4.Ссылки.

Файлы устройств представляют собой устройства компьютера. Файлы устройств находятся в каталоге /dev. Например, /dev/ttySO первый последовательный порт (COM1). Обычные файлы, в свою очередь, делятся на нормальные (текстовые) и двоичные.

Каталоги - это специальные файлы, содержащие информацию о других файлах (файлах устройств (/dev), обычных файлов и ссылок). Конечно, это довольно грубое определение, скорее интуитивное, чем точное.

Ссылки позволяют хранить один и тот же файл, но под разными именами.

Максимальная длина имени файла составляет 254 символа. Имя может содержать практически любые символы, кроме: / \ ? > < | " *

Свойства файловой системы ext2:

- Максимальный размер файловой системы 4 Тбайт.
- Максимальный размер файла 2 Гбайт.
- Максимальная длина имени файла 255 символов.
- Минимальный размер блока 1024 байт.
- Количество выделяемых индексных дескрипторов 1 на 4096 байт раздела.

***2 Команды для работы с файлами и каталогами***

**2.1 Команды для работы с файлами**

Прежде чем приступить к описанию команд для работы с файлами, необходимо отметить, что для выполнения операций над файлами вы должны иметь права доступа к этим файлам. О правах доступа будет написано ниже, а здесь предполагается, что пользователь имеет права доступа к используемым файлам.

Создание и просмотр файла

Для просмотра файла обычно используется команда **cat**.

Например:**cat file.txt**

При этом на стандартный вывод, то есть на ваш терминал, будет выведен файл file.txt. Однако более удобными командами для просмотра файлов являются команды **more** или **less**:

Практически все операционные системы обладают механизмом перенаправления ввода/вывода, и Linux не является исключением из этого правила.

Обычно программы вводят текстовые данные с консоли (терминала) и выводят данные на консоль. При вводе под консолью подразумевается клавиатура, а при выводе - экран монитора. Клавиатура и экран монитора - это, соответственно, стандартный ввод и вывод (stdin и stdout). Любой ввод/вывод можно интерпретировать как ввод из некоторого файла и вывод в файл. Работа с файлами производится через их дескрипторы. Для организации ввода/вывода в UNIX используются три файла: stdin (дескриптор 0), stdout (дескриптор 1) и stderr (дескриптор 2).

Символ > («больше») используется для перенаправления стандартного вывода в файл. Например:

**cat > newfile.txt**

В этом примере стандартный вывод команды cat будет перенаправлен в файл newfile.txt, который будет создан после выполнения этой команды. Если файл с этим именем уже существует, то он будет перезаписан.

Здесь используется перенаправление ввода/вывода. Данные со стандартного ввода (клавиатуры) перенаправляются в файл file.txt. Проще говоря все, что вы после этой команды введете с клавиатуры, будет записано в файл file.txt. Нажатие **Ctrl + D** остановит перенаправление и прервет выполнение команды **cat**.

Символ < («меньше») используется для переназначения стандартного ввода команды. Например, при выполнении команды cat < file.txt в качестве стандартного ввода будет использован файл file.txt, а не клавиатура. Символ >> используется для присоединения данных в конец файла (append) стандартного вывода команды. Например, в отличие от случая с символом >, выполнение команды **cat >> newfile.txt** не перезапишет файл в случае его существования, а добавит данные в его конец.

Помните, что вы не сможете создать файл в каталоге, к которому у вас нет доступа. Вы даже не сможете просмотреть файл, если пользователь, которому этот файл принадлежит, запретил чтение этого файла.

Копирование файла

Для копирования файлов в ОС Linux используется команда **cp**, которая имеет следующий формат вызова:

cp [параметры] источник назначение

Рассмотрим несколько примеров:

**cp file.txt file2.txt**

**cp file.txt /home/user/text/**

В первом случае выполняется копирование файла file.txt в файл file2.txt. Оба файла находятся в текущем каталоге. Во втором случае копирование файла file.txt в каталог /home/user/text/.

Вы можете использовать точку (.) в качестве ссылки на текущий каталог, символ тильды ( ~ ) - на домашний каталог. Родительский каталог обозначается двумя точками (..). Корневой каталог обозначается символом косой черты (/). Параметры команды cp указаны в табл. 1

Таблица 1

| Параметр | Описание |
|---|---|
| -a | При копировании сохраняются атрибуты файлов |
| -b | Создание копии вместо перезаписи существующего файла |
| -d | Поддержка символических ссылок. При этом копироваться будут сами символические ссылки без файлов, на которые они указывают |
| -i | Перед перезаписью существующего файла от пользователя потребуется подтверждение этого |
| -l | Создание жестких ссылок вместо копирования (при копировании в каталог) |
| -r | Копирование каталога вместе с подкаталогами |
| -s | Создание символических ссылок вместо копирования (при копировании в каталог) |
| -u | Не перезаписывать, если перезаписываемый файл имеет более позднюю дату модификации |
| -v | Вывод сведений обо всех выполняемых действиях (verbose). Выводит имена всех копируемых файлов |
| -x | Игнорировать каталоги, расположенные в других файловых системах, по отношению к системе, откуда выполняется копирование |

Удаление файла и каталога

Для удаления указанного файла используется команда **rm**. Например: **rm ffle2.txt**. При этом для удаления файла пользователь должен иметь право на запись в каталог. Права на чтение или запись файла необязательны. Если нет права на запись в файл, то выдается (в восьмеричном виде) режим доступа к файлу и запрашивается подтверждение на удаление. Если стандартный вывод назначен не на терминал, то команда **rm** будет вести

себя так же, как при наличии опции **-f**. А при указании опции **-f** не выдается предупреждений, если удаляемый файл не существует, а также не запрашивается подтверждение при удалении файла, на запись в который нет прав. Если нет права и на запись в каталог, то файлы не удаляются. Сообщение об ошибке выдается лишь при попытке удалить каталог, на запись в который нет прав.

Опция **-r** предназначена для рекурсивного удаления всех файлов и каталогов, указанных в командной строке. При удалении непустых каталогов команда **rm** с параметром **-r** предпочтительнее, чем команда **rmdir**, поскольку последняя не может удалить непустой каталог.

Опция **-i** аналогична одноименной опции команды **cp** и требует подтверждения от пользователя перед удалением каждого файла.

Быстрый поиск файла

Команда **locate** производит поиск заданного файла в файловой системе. Вместо имени файла можно указать образец имени, например, в тех случаях, когда вы забыли точное название файла:

**locate passwd**

Поиск программы

Если вы не знаете, в каком каталоге находится нужная вам команда (программа), введите команду **which**, указав в качестве параметра нужную вам команду (программу).

**which awk**

Данная команда бывает очень полезна в тех случаях, когда вы хотите узнать, установлена ли вообще та или иная программа. Быстрый поиск имени программы можно выполнить прямо из командной строки Linux: для этого введите первые буквы нужной вам команды и нажмите «Tab». Такая функция называется автозаполнением командной строки. Для вывода всех доступных команд нажмите «Tab» дважды. Естественно, что полный список команд на одном экране не поместится. Чтобы «листать» консоль, используйте клавиши «PageUp» и «PageDown».

**2.2. Команды для работы с каталогами**

Просмотр содержимого каталога. Для просмотра содержимого каталога используется команда **ls**. Закоренелые пользователи DOS могут использовать привычную им команду **dir**, но команда **ls** намного удобнее. Программа **ls** имеет около сорока параметров, о назначении которых вы можете узнать в справочной системе, введя команду **man ls**.

Вывод имени текущего каталога. Команда **pwd** сообщит вам имя текущего каталога. Эту команду очень удобно использовать при написании сценариев.

Файловый менеджер Midnight Commander. Для вызова Midnight Commander введите команду **mc**. Естественно, пакет mc должен быть предварительно установлен. Midnight Commander очень похож на всем известный Norton Commander.

### *3. Стандартные имена устройств в Linux*

Как уже отмечалось раньше, все устройства в Linux являются файлами. Файлы устройств находятся в специальном каталоге /dev. Для просмотра данного каталога удобнее всего использовать команду mc. Запустите mc и перейдите в каталог /dev. Если возле файла слева вы видите +, то данное устройство подключено и функционирует.

Определим какие файлы ассоциируются с какими устройствами. Договоримся, что символ N обозначает номер устройства, например, ttyN обозначает устройства /dev/ttyl .. /dev/ttyN, а x - символ. Наиболее используемые стандартные имена устройств (в соответствии с принятыми обозначениями) приведены в табл. 2. На устройствах hdxN и sdxN необходимо остановиться подробнее. Известно, что к (E)IDE (ATA) контроллеру можно подключить четыре IDE устройства: Primary Master, Primary Slave, Secondary Master, Secondary Slave.

Таблица 2 Наиболее используемые стандартные имена устройств

| Файл | Устройство |
|---|---|
| TtyN | Консоль |
| mouse | Мышь |
| audio | Звуковая карта |
| modem | Модем. Обычно файл /dev/modem является ссылкой на один из файлов /dev/ttySO |
| ttySN | Последовательный порт. Файл /dev/ttySO аналогичен файлу COM4 в DOS |
| IpN | Параллельный порт |
| cuaN | Могут обозначать последовательные порты. Используются немного в другом контексте, чем ttySN |
| hdxN | IDE жесткий диск |
| sdxN | SCSI жесткий диск |
| fdO | Первый дисковод для гибких дисков, то есть A:, для B: используется имя /dev/fd1 |
| stN | Стример с интерфейсом SCSI |
| nrtfN | Стример с интерфейсом FDC |
| mdN | Массив RAID |
| ethN | Сетевая плата |
| null | Пустое устройство |

Этим устройствам соответствуют символы: a, b, c, d. Например, /dev/hda -Primary Master, a /dev/hdd - Secondary Slave. Номер N в обозначении устройства обозначает номер

раздела на жестком диске. Первичный раздел DOS на первом жестком диске обозначается так: /dev/hdal.

***4. Стандартные каталоги***

В ОС Linux есть каталоги, которые называются стандартными. Иногда их еще называют системными. Эти каталоги присутствуют практически в каждой ОС Linux. В них находятся файлы, необходимые для управления и сопровождения системы, а также стандартные программы. Описание стандартных каталогов сведено в табл. 3.

Таблица 3 Стандартные каталоги

| Каталог | Назначение |
|---|---|
| / | Корневой каталог |
| /bin | Содержит стандартные программы |
| /home | Содержит домашние каталоги пользователей |
| /usr | Содержит все программы, используемые системой |
| /sbin | Команды для системного администрирования |
| /var | Содержит файлы, которые постоянно изменяются, например, а также файлы почтовых ящиков спулы для принтеров, |
| /etc | Содержит файл настройки системы |
| /dev | Здесь находятся файлы устройств |
| /tmp | Каталог для временных файлов |
| /mnt | Обычно здесь создаются точки монтирования. Тем не менее, подмонтировать файловую систему можно к любому другому каталогу, а использование каталога /mnt не является обязательным |

***5. Создание файловой системы. Типы файловых систем***

**5.1 описание файловых систем**

Каждая операционная система имеет основной тип файловой системы, а также дополнительные типы, поддержка которых осуществляется модулями (драйверами), подключаемыми к ядру. В случае с Linux поддержку той или иной файловой системы можно встроить непосредственно в ядро. Основной файловой системой Linux на момент написания этих строк является ext2fs, однако на ее смену сейчас приходит ext3fs и последние версии дистрибутивов Linux используют именно ее. Переход на новую файловую систему обеспечивает более надежную ее работу.

Кроме основной файловой системы, Linux поддерживает файловые системы, указанные в табл.4.

В табл. 4 рассмотрены базовые типы файловых систем. ОС Linux поддерживает и другие файловые системы, не указанные в таблице. Поддержку нужной вам файловой

системы можно включить при перекомпилировании ядра. Для нормальной работы вам потребуются только файловые системы, отмеченные звездочкой.

Таблица 4 Типы файловых систем

| Файловая система | Комментарий |
|---|---|
| Minix Filesystem (minix) | Устаревшая и практически неиспользуемая в наше время файловая система |
| Xia Filesystem (xia) | Редко используется |
| UMSDOS Filesystem (umsdos) | Использовалась для установки Linux в раздел MSDOS |
| MSDOS Filesystem (msdos) | Файловая система msdos |
| (*) VFAT Filesystem (vfat) | Файловая система Windows 9x |
| NT Filesystem (ntfs) | Файловая система Windows NT |
| HPFS Filesystem (hpfs) | High Performance FS. Файловая система OS/2 |
| (*) ISO 9660 | Файловая система, используемая большинством CDROM |
| (*) /proc | Предоставляет информацию о процессах |
| Extended Filesystem (ext) | Устаревшая версия основной файловой системы Linux |
| (*) Second Extended Filesystem (ext2) | Основная файловая система |
| или Third Extended Filesystem (ext3) | Основная файловая система |
| Network Filesystem (nfs) | Сетевая файловая система |

Ядро 2.4.8 уже поддерживает файловые системы: Ext3, ReiserFS, XFS.

Список файловых систем, которые поддерживаются ядром системы, содержится в файле **/proc/filesystems**. Просмотреть этот список поможет команда **cat /proc/filesystems.**

Файловая система Ext3 (Third Extended Filesystem) представляет собой журналируемую надстройку над ext2, поэтому возможно чтение одной файловой системы как драйвером Ext3, так и драйвером Ext2. Возможно отключение журналирования. Файловую систему ext2 можно конвертировать в ext3, запустив программу создания журнала. После конвертирования новую файловую систему можно использовать и без журнала - для этого достаточно примонтировать ее драйвером для ext2.

RaiserFS - журналируемая файловая система. Основной ее особенностью является способность хранить несколько мелких файлов в одном блоке.

XFS - также журналируемая файловая система, первоначально разрабатывалась компанией Silicon Graphics (SGI) для ОС Irix. Особенностью этой файловой системы является устройство журнала: в журнал пишется часть метаданных самой файловой системы таким образом, что весь процесс восстановления после сбоя сводится к

копированию этих данных из журнала в файловую систему. Размер журнала задается при создании системы, он должен быть не меньше 32 мегабайт.

JFS первоначально разрабатывалась компанией IBM для AIX OS, позднее была перенесена на OS/2, а не так давно и под Linux. Размер журнала составляет примерно 40% от размера файловой системы. Максимальный размер равен 32 мегабайтам. Эта файловая система может содержать несколько сегментов, содержащих журнал и данные. Эти сегменты называются агрегатами и могут монтироваться отдельно.

Все эти файловые системы предназначены для создания высокопроизводительного файлового сервера или рабочей станции, ориентированной на работу с файлами больших размеров. Какая из них лучше - трудно сказать. Нужно исходить из потребностей. Производительность JFS ниже, чем у остальных трех файловых систем, но она более предсказуема по своему поведению, то есть можно с большой вероятностью предсказать, когда начнется падение производительности. XFS обладает значительно большими показателями производительности. Особенно хорошо она себя проявляет при работе с файлами больших размеров. Производительность этой файловой системы можно значительно повысить, если создать журнал на отдельном контроллере.

Файловая система ReiserFS показала еще большую производительность, но трудна в прогнозировании падения производительности. Файловая система ext3 практически по всем параметрам производительности мало чем отличается от ReiserFS.

Давайте же разберемся, что же собой представляет журналируемая файловая система, и в чем состоят ее преимущества. Прежде всего нужно отметить, что журналируемые файловые системы не предназначены для восстановления ваших данных любой ценой после сбоя. Они предназначены для других целей. Например, вы открываете файл, и он успешно открывается - файловая система отмечает операцию открытия в своем журнале записью транзакции. Затем вы начинаете писать в файл. При этом файловая система не запоминает копии этих данных. Затем происходит сбой. Когда происходит восстановление после сбоя, происходит откат до последней успешной транзакции - открытия нового пустого файла. Поэтому, когда вы пишите в файл и происходит сбой, вы получите файл нулевой длины.

Файловая система ext3 имеет два основных преимущества перед ext2. Первое состоит в том, что ext3 записывает изменение данных и метаданных, что позволяет сохранять содержимое файлов. Второе преимущество состоит в том, что разделы ext3 ничем не отличаются от разделов ext2, поэтому всегда можно перейти к старой файловой системе и наоборот. Главным здесь является то, что вы можете спокойно делать резервную

копию файловой системы ext3, а потом развернуть ее на ext2-разделе. Позже можно будет включить журналирование.

Немного определений:

Метаданные (metadata) -- это данные, которые являются описанием других данных (например, схема базы данных по отношению к содержимому базы данных).

Журналирование - это запись изменения метаданных во время совершения транзакции. В журнал записывается состояние трех типов данных: метаданных, блоков описания и блоков-заголовков. Уровень журналирования (то, что будет писаться в журнал) можно указать при монтировании файловой системы с помощью программы mount. Журналируемый блок всегда записывается полностью, даже если произошло маленькое изменение. Это делается очень быстро, так как операции журналируемого ввода/вывода объединены в большие кластеры.

Блоки описания описывают другие метаданные. Запись блоков описания происходит перед записью метаданных. Блоки-заголовки описывают заголовок и окончание журнала. Каждому блоку заголовку присваивается порядковый номер, чтобы гарантировать упорядоченную запись во время восстановления.

### 5.2 Программа diskdrake

Программа diskdrakerake обладает понятным графическим интерфейсом и запускается из-под X Window. Эта программа входит в состав инсталлятора Linux Mandrake, и именно ее вы используете, когда формируете разделы на вашем винчестере при установке этого дистрибутива. По своим возможностям она очень напоминает Partition Magic, да и интерфейс мало чем отличается. Вы можете создавать и удалять разделы, изменять размер и тип файловой системы. В состав Red Hat Linux входит всем известный DiskDruid. Программа похожа на diskdrake, но, на мой взгляд, менее удобна.

### 5.3 Монтирование и размонтирование разделов

Для монтирования ФС предназначена программа **mount**, для размонтирования - **umount**. Общий формат вызова (наиболее часто используемый) следующий:

**mount -t fs_type device mount_point**

В качестве параметра **fs_type** программы **mount** указывается тип подключаемой файловой системы, некоторые из которых я позволю себе еще раз напомнить в табл. 5.

Таблица 5

| Тип | Описания |
|---|---|
| ext2 или ext3 | Файловая система Linux |
| Vfat | Файловая система Windows Эх |

| | |
|---|---|
| iso9660 | Ее нужно использовать при монтировании CD-ROM |
| Ntfs | Всем известная NT Filesystem |

Следующим параметром является устройство (**device**). В качестве устройства выступает носитель данных, например **/dev/hdd**. Далее, наконец, задается сама точка монтирования (mountjoint). Примонтировать файловую систему вы можете к любому каталогу корневой файловой системы.. Например, для монтирования дисковода A: вы можете использовать следующую команду:

**mount -t vfat /dev/fdO /mnt/flopyy**

При этом считается, что дискета отформатирована для файловой системы vfat.

Для монтирования привода CD-ROM вы можете воспользоваться следующей командой**:**

**mount -t iso9660 /dev/hdd /mnt/cdrom**

Привод CD-ROM подключен ко второй шине IDE как ведомый (Secondary Slave).

Для размонтирования достаточно указать точку монтирования или устройство в качестве параметра команды **umount**. Например, команда **umount /rant/floppy** размонтирует диск A:.

Информация об устройствах, смонтированных на данный момент, содержится в файле **/etc/mtab**.

Программа **mount** имеет опции, представленные в табл. 6.

Таблица 6 Параметры программы mount

| Опция | Описания |
|---|---|
| -a | Монтирование всех файловых систем, указанных в файле /etc/fstab, кроме тех,для которых указан параметр noauto |
| -n | Монтирование без записи в файл /etc/mtab. Полезно, если каталог /etc доступен толькодля чтения |
| -r | Монтирование в режиме «только чтение» |
| -w | Монтирование в режиме «чтение/запись» (по умолчанию) |
| -t тип ФС | Задает тип файловой системы |

Вы можете комбинировать опции, например, команда **mount -a -t vfat** монтирует все ФС типа VFAT. Список файловых систем, которые поддерживает ядро вашей системы, находится в файле /etc/filesystems или в файле /proc/filesystems.

Для того, чтобы файловая система монтировалась автоматически при загрузке системы, нужно внести определенную запись в файл /etc/f stab. Формат записей в этом файле следующий:

**device mount_point fs_type options флаг_реэервного_копирования флаг_проверки**

где:

**device** - устройство, которое нужно подмонтировать;

**mount_point** - точка монтирования;

**fs_type** - тип файловой системы;

**options** - набор опций монтирования (см. табл. 7);

**флаг_резервного_копирования** - если установлена (1), то программа dump включит данную ФС в архив при создании резервной копии (дампа). Если установлен (0), то резервная копия ФС создаваться не будет;

**флаг_проверки** - этот флаг устанавливает порядок, в котором файловые системы при монтировании будут проверяться на наличие ошибок. Поиск и исправление ошибок при этом осуществляется специальной программой **fsck**, которая запускается сценарием инициализации системы. Этот флаг означает очередь, в которой будет проверяться данная файловая система.

Если для нескольких файловых систем указан один и тот же номер, то эти файловые системы, при подходе очереди, будут проверяться одновременно.

Правильная настройка флагов проверки позволяет ускорить загрузку. Корневая файловая система всегда должна иметь значение флага проверки (1), которое означает, что ее необходимо проверять первой. Для всех остальных файловых систем рекомендуется устанавливать значение (2), которое позволит произвести их проверку одновременно, сразу же после проверки корневой файловой системы. Значение (0) указывается для файловых систем, проверку которых производить не нужно. К таким ФС относятся съемные файловые системы (носители Floppy, CD-ROM, и т.д.).

Таблица 7 Опции монтирования ФС в файле /etc/fstab

| Опция | Описание |
|---|---|
| exec | Разрешает запуск бинарных (выполняемых) файлов для данной файловой системы. Эта опция используется по умолчанию |
| noexec | Запрещает запуск бинарных (выполняемых) файлов для данной файловой системы |
| noauto | Данная файловая система не может быть смонтирована с помощью команды mount –a, то есть не может быть смонтирована при загрузке системы |
| auto | Данная файловая система будет автоматически смонтирована во время загрузки. Эта опция используется по умолчанию |
| ro | Монтирование в режиме «только чтение» |
| rw | Монтирование в режиме «чтение/запись». Эта опция используется по |

| | |
|---|---|
| | умолчанию |
| user | Разрешает пользователям монтировать/демонтировать данную файловую систему |
| nouser | Запрещает пользователям монтировать/демонтировать данную файловую систему. Эта опция используется по умолчанию |
| defaults | Использовать стандартный набор опций, установленных по умолчанию |

Опцию поехес полезно устанавливать для файловых систем, в которых вы не предполагаете запускать программы. Ее полезно установить для файловой системы vfat: запускать там нечего, а вот при копировании из нее файлов в файловую систему ext2 не будет устанавливаться право на выполнение только что скопированного файла. Если вы установите опцию noauto, данную систему нельзя будет подмонтировать с помощью опции -а программы mount. Команда mount -а обычно выполняется при запуске системы, а значит, данная файловая система не будет подмонтирована автоматически. Это очень полезно для сменных устройств, например, дискет или магнитооптических дисков, когда нужно просто задать какие-нибудь параметры для данной файловой системы, но не монтировать ее. Ведь при запуске системы в приводе может не оказаться дискеты или магнитооптического диска. Опция user позволяет пользователю монтировать данную файловую систему. Обычно она используется вместе с опцией noauto для сменных дисков. Пример файла конфигурации файловых систем /etc/fstab приведен в следующем листинге.

```
/dev/hdal / ext2 defaults 1 1
/dev/hda2 /den ext2 defaults 0 2
/dev/hda3 /home ext2 defaults 0 2
/dev/hda4/swap swap defaults O O
/dev/fdO /mt/floppy vfat noauto,поехес O O
/dev/hdd/mt/cdrom iso9660 noauto,го O O
none /proc proc defaults O O
```

В первой строке содержится запись, задающая параметры монтирования корневого раздела «/» и указывающая, что устройство /dev/hda1 имеет файловую систему ext2 и должно быть смонтировано со стандартным набором опций defaults, используемых по умолчанию. Кроме этого, в записи сказано, что необходимо создавать резервную копию данной файловой системы, и что устройство должно быть проверено на наличие ошибок при загрузке системы, причем в первую очередь.

Вторая и третья записи содержат информацию о том, что устройства /dev/hda2 и /dev/hda3 содержат файловую систему ext2 и должны быть смонтированы со стандартными установками в каталоги /den и /home соответственно. Резервные копии

данных файловых систем создавать не нужно, а проверку при загрузке ОС необходимо производить во вторую очередь, причем одновременно обеих.

Четвертая строка содержит запись о параметрах монтирования раздела подкачки (swap). Для этого, а также для всех последующих разделов указано, что не надо ни создавать их резервную копию, ни производить их проверку при загрузке.

В пятой и шестой строках монтируются устройство чтения дискет (Floppy) и CD-ROM. Последняя строка файла /etc/fstab определяет специальную файловую систему /proc, которой вообще не ставится в соответствие никакое устройство (none). Файловая система /proc предназначена для обеспечения интерфейса взаимодействия с внутренними структурами данных ядра.

В процессе настройки вы, наверное, заметите, что при монтировании файловой системы vfat вместо русских букв отображается не совсем то, что вам бы хотелось. Например в лучшем случае вместо имени каталога Мои документы вы увидите ??? ??????????. Для перекодирования русскоязычных (и не только) имен файлов из одной кодировки в другую используются опции монтирования

iocharset и codepage. Непосредственно для vfat нужно указать: codepage=866, iocharset=koi8-r

## Задание

1. создайте в вашем домашнем каталоге файл report.txt.
2. выясните поддерживает ли ваша ОС файловую систему ext3.
3. запишите отчет в файл report.txt, предварительно создав заголовок путём ввода с клавиатуры в виде:

**********
Supported types of file systems
---
{типы файловых систем}
**********

4. выключите систему набрав в консоли команду halt или shutdown.
5. в разделе Settings программы Microsoft Virtual PC создайте виртуальный жесткий диск ( ≈500 мб).
6. подключите его к виртуальной машине на контроллер Primary Slave.
7. загрузите ОС linux.
8. получите информацию об имеющихся разделах и точках их монтирования.
9. запишите эти данные в файл report.txt в виде

**********

Existed partitions

step1

---

{существующие разделы и

точки их монтирования}

**********

**10.** с помощью команды locate найдите и запустите программу для создания и изменения разделов на жестком диске.

**11.** создайте на новом жёстком диске два раздела, равных по объему.

**12.** отформатируйте полученные разделы в ФС ext3 и xfs.

**13.** создайте в корневом каталоге системы каталог mount_fs, создайте в нем каталоги part1 и part2.

**14.** получите права суперпользователя, набрав команду **su.**

**15.** примонтируйте раздел с ФС ext3 второго жесткого диска к каталогу part1, и соответственно раздел с ФС xfs к каталогу part2.

**16.** получите информацию об имеющихся разделах и точках их монтирования.

**17.** запишите эти данные в файл report.txt в виде

**********

Existed partitions

Step2

---

{существующие разделы и

точки их монтирования}

**********

**18.** создайте в домашнем каталоге каталог part3.

**19.** размонтируйте раздел с ФС ext3. Удалите каталог, к которому он был примонтирован.

**20.** примонтируйте данный раздел к каталогу /home/part3

**21.** получите информацию об имеющихся разделах и точках их монтирования.

**22.** проверьте полученные разделы на наличие ошибок.

**23.** запишите эти данные в файл report.txt в виде

**********

Existed partitions

Step3

---

{существующие разделы и

точки их монтирования}

**********

**24.** установите опцию автоматического монтирования этого раздела при загрузке

**25.** данные файла конфигурации также добавьте в файле report.txt, с соответствующим заголовком.

## **Лабораторная работа № 7.**

## Организация ввода-вывода в UNIX

Современные версии UNIX-подобных операционных систем умеют работать с разнообразными файловыми системами, различающимися своей организацией. Такая возможность достигается с помощью разбиения каждой файловой системы на зависимую и независимую от конкретной реализации части, подобно тому, как в лекции 13, посвященной вопросам ввода-вывода, мы отделяли аппаратно-зависимые части для каждого устройства – *драйверы* – от общей базовой подсистемы ввода-вывода. Независимые части всех файловых систем одинаковы и представляют для всех остальных элементов ядра абстрактную файловую систему, которую принято называть ***виртуальной файловой системой***. Зависимые части для различных файловых систем могут встраиваться в ядро на этапе компиляции, либо добавляться к нему динамически по мере необходимости, без перекомпиляции системы (как в системах с микроядерной архитектурой).

Рассмотрим схематично устройство *виртуальной файловой системы*. В файловой системе s5fs данные о физическом расположении и атрибутах каждого открытого файла представлялись в операционной системе структурой данных в таблице индексных узлов открытых файлов (см. семинар 11–12, раздел «Системные вызовы и команды для выполнения операций над файлами и директориями»), содержащей информацию из индексного узла файла во вторичной памяти. В *виртуальной файловой системе*, в отличие от s5fs, каждый файл характеризуется не индексным узлом inode, а некоторым *виртуальным узлом vnode*. Соответственно, вместо таблицы индексных узлов открытых файлов в операционной системе появляется *таблица виртуальных узлов открытых файлов*. При открытии файла в операционной системе для него заполняется (если, конечно, не был заполнен раньше) элемент *таблицы виртуальных узлов открытых файлов*, в котором хранятся, как минимум, тип файла, счетчик числа открытий файла, **указатель** на реальные физические данные файла и, **обязательно, указатель** на таблицу системных вызовов, совершающих операции над файлом, – *таблицу операций*. Реальные физические данные файла (равно как и способ расположения файла на диске и т.п.) и системные вызовы, реально выполняющие операции над файлом, уже не являются элементами *виртуальной файловой системы*. Они относятся к одной из зависимых частей файловой системы, так как определяются ее конкретной реализацией.

При выполнении операций над файлами по *таблице операций*, чей адрес содержится в *vnode*, определяется системный вызов, который будет на самом деле выполнен над реальными физическими данными файла, чей адрес также находится в *vnode*. В случае с s5fs данные, на которые ссылается *vnode*, – это как раз данные индексного узла, рассмотренные на семинарах 11–12 и на лекции 12. Заметим, что *таблица операций* является общей для всех файлов, принадлежащих одной и той же файловой системе.

**Операции над файловыми системами. Монтирование файловых систем**

В материалах семинаров 11–12 рассматривалась только одна файловая система, расположенная в одном разделе физического носителя. Как только мы переходим к сосуществованию нескольких файловых систем в рамках одной операционной системы, встает вопрос о логическом объединении структур этих файловых систем. При работе операционной системы нам изначально доступна лишь одна, так называемая корневая, файловая система. Прежде, чем приступить к работе с файлом, лежащим в некоторой другой файловой системе, мы должны встроить ее в уже существующий ациклический граф файлов. Эта операция – операция над файловой системой – называется *монтированием файловой системы* (mount).

Для *монтирования файловой системы* (см. лекцию 12, раздел «*Монтирование файловых систем*») в существующем графе должна быть найдена или создана некоторая пустая директория – точка монтирования, к которой и присоединится корень монтируемой файловой системы. При операции монтирования в ядре заводятся структуры данных, описывающие файловую систему, а в *vnode* для точки монтирования файловой системы помещается специальная информация.

*Монтирование файловых систем* обычно является прерогативой системного администратора и осуществляется командой операционной системы *mount* в ручном режиме, либо автоматически при старте операционной системы. Использование этой команды без параметров не требует специальных полномочий и позволяет пользователю получить информацию обо всех смонтированных файловых системах и соответствующих им физических устройствах. Для пользователя также обычно разрешается *монтирование файловых систем*, расположенных на

гибких магнитных дисках. Для первого накопителя на гибких магнитных дисках такая команда в Linux будет выглядеть следующим образом:

mount /dev/fd0 <имя пустой директории>

где <имя пустой директории> описывает точку монтирования, а /dev/fd0 – специальный файл устройства, соответствующего этому накопителю (о специальных файлах устройств будет подробно рассказано в следующем разделе).

Если мы не собираемся использовать смонтированную файловую систему в дальнейшем (например, хотим вынуть ранее смонтированную дискету), нам необходимо выполнить операцию логического разъединения смонтированных файловых систем (umount). Для этой операции, которая тоже, как правило, является привилегией системного администратора, используется команда *umount* (может выполняться в ручном режиме или автоматически при завершении работы операционной системы). Для пользователя обычно доступна команда отмонтирования файловой системы на дискете в форме

umount <имя точки монтирования>

где <имя точки монтирования> – это <имя пустой директории>, использованное ранее в команде *mount*, или в форме

umount /dev/fd0

где /dev/fd0 – специальный файл устройства, соответствующего первому накопителю на гибких магнитных дисках.

**Заметим, что для последующей корректной работы операционной системы при удалении физического носителя информации обязательно необходимо предварительное логическое разъединение файловых систем, если они перед этим были объединены.**

**Блочные, символьные устройства. Понятие драйвера. Блочные, символьные драйверы, драйверы низкого уровня. Файловый интерфейс**

Обремененные знаниями об устройстве современных файловых систем в UNIX, мы можем, наконец, заняться вопросами реализации подсистемы ввода-вывода.

В лекции 13 (раздел «Структура системы ввода-вывода») речь шла о том, что все устройства ввода-вывода можно разделить на относительно небольшое число типов, в зависимости от набора операций, которые могут ими выполняться. Такое деление позволяет организовать «слоистую» структуру подсистемы ввода-вывода, вынеся все аппаратно-зависимые части в *драйверы устройств*, с которыми взаимодействует базовая подсистема ввода-вывода, осуществляющая стратегическое управление всеми устройствами.

В операционной системе UNIX принята упрощенная классификация устройств (см. лекцию 13, раздел «Систематизация внешних устройств и интерфейс между базовой подсистемой ввода-вывода и *драйверами*»): все устройства разделяются по способу передачи данных на *символьные* и *блочные*. ***Символьные устройства*** осуществляют передачу данных байт за байтом, в то время как ***блочные устройства*** передают блок байт как единое целое. Типичным примером *символьного устройства* является клавиатура, примером *блочного устройства* – жесткий диск. Непосредственное взаимодействие операционной системы с устройствами ввода-вывода обеспечивают их *драйверы*. Существует пять основных случаев, когда ядро обращается к *драйверам*:

1. Автоконфигурация. Происходит в процессе инициализации операционной системы, когда ядро определяет наличие доступных устройств.
2. Ввод-вывод. Обработка запроса ввода-вывода.
3. Обработка прерываний. Ядро вызывает специальные функции *драйвера* для обработки прерывания, поступившего от устройства, в том числе, возможно, для планирования очередности запросов к нему.
4. Специальные запросы. Например, изменение параметров *драйвера* или устройства.
5. Повторная инициализация устройства или останов операционной системы.

Так же как устройства подразделяются на *символьные* и *блочные*, *драйверы* тоже существуют символьные и блочные. Особенностью *блочных устройств* является возможность организации на них файловой системы, поэтому блочные *драйверы* обычно используются файловой системой UNIX. При обращении к *блочному устройству*, не содержащему файловой системы, применяются специальные *драйверы* низкого уровня, как правило, представляющие собой интерфейс между ядром операционной системы и блочным *драйвером* устройства.

Для каждого из этих трех типов *драйверов* были выделены основные функции, которые базовая подсистема ввода-вывода может совершать над устройствами и *драйверами*:

инициализация устройства или *драйвера*, временное завершение работы устройства, чтение, запись, обработка прерывания, опрос устройства и т.д. (об этих операциях уже говорилось в лекции 13, раздел «Систематизация внешних устройств и интерфейс между базовой подсистемой ввода-вывода и *драйверами*»). Эти функции были систематизированы и представляют собой интерфейс между *драйверами* и базовой подсистемой ввода-вывода.

Каждый *драйвер* определенного типа в операционной системе UNIX получает собственный номер, который по сути дела является индексом в массиве специальных структур данных операционной системы – ***коммутаторе устройств*** соответствующего типа. Этот индекс принято также называть ***старшим номером устройства***, хотя на самом деле он относится не к устройству, а к *драйверу*. Несмотря на наличие трех типов *драйверов*, в операционной системе используется всего два *коммутатора*: для блочных и символьных *драйверов*. *Драйверы* низкого уровня распределяются между ними по преобладающему типу интерфейса (к какому типу ближе – в такой массив и заносятся). Каждый элемент *коммутатора устройств* обязательно содержит адреса (точки входа в *драйвер*), соответствующие стандартному набору функций интерфейса, которые и вызываются операционной системой для выполнения тех или иных действий над устройством и/или *драйвером*.

Помимо *старшего номера устройства* существует еще и ***младший номер устройства***, который передается *драйверу* в качестве параметра и смысл которого определяется самим *драйвером*. Например, это может быть номер раздела на жестком диске (partition), доступ к которому должен обеспечить *драйвер* (надо отметить, что в операционной системе UNIX различные разделы физического носителя информации рассматриваются как различные устройства). В некоторых случаях *младший номер устройства* может не использоваться, но для единообразия он должен присутствовать. Таким образом, пара драйвер-устройство всегда однозначно определяется в операционной системе заданием пары номеров (*старшего* и *младшего номеров устройства*) и типа *драйвера* (символьный или блочный).

Для связи приложений с *драйверами устройств* операционная система UNIX использует **файловый интерфейс**. В числе типов файлов на предыдущем семинаре упоминались специальные файлы устройств. Так вот, каждой тройке тип-драйвер-устройство в файловой системе соответствует специальный файл устройства, который не занимает на диске никаких логических блоков, кроме индексного узла. В качестве атрибутов этого файла помимо обычных атрибутов используются соответствующие *старший* и *младший номера устройства* и тип *драйвера* (тип драйвера определяется по типу файла: ибо есть специальные файлы *символьных устройств* и специальные файлы *блочных устройств*, а номера устройств занимают место длины файла, скажем, для регулярных файлов). Когда открывается специальный файл устройства, операционная система, в числе прочих действий, заносит в соответствующий элемент *таблицы открытых виртуальных узлов* указатель на набор функций интерфейса из соответствующего элемента *коммутатора устройств*. Теперь при попытке чтения из файла устройства или записи в файл устройства *виртуальная файловая система* будет транслировать запросы на выполнение этих операций в соответствующие вызовы нужного *драйвера*.

Мы не будем останавливаться на практическом применении файлового интерфейса для работы с устройствами ввода-вывода, поскольку это выходит за пределы нашего курса, а вместо этого приступим к изложению концепции *сигналов* в UNIX, тесно связанных с понятиями *аппаратного прерывания*, *исключения* и *программного прерывания*.

Литература:


1. Основы работы с операционной системой WINDOWS 9x:Метод.указ. по дисц. "Информатика" для индив. и самост. работы студ. педспец./Койбагарова Т.К.-Павлодар:ПГУ им.С.Торайгырова,2002
2. Острейковский В.А. Информатика:Учеб. для техн.вузов.-М.:Высш.шк.,1999
3. Партыка Т. Л. , Попов И.И. Операционные системы, среды и оболочки: Учеб. пособие.- М. :Форум: Инфра-М,2003
4. Грибанов В. П. и др. Операционные системы:Учеб.пособие для вузов/С.В.Дробин,В.Д.Медведев.-М.:Финансы и статистика,1990
5. Пасько В., Колесников А. Самоучитель работы на персональном компьютере.- 2- е изд.,доп.-К.:Издат. группа BHV,1999

6. Лорин Г. , Дейтел Х.М. Операционные системы.-М.:Финансы и статистика,1984