Методические указания к лабораторным работам

по дисциплине «**Операционные системы**»

Часть 1

Работы выполняются в среде операционных систем Windows и. для выполнения работ необходимо использовать приведенную в библиографическом списке литературу, конспект курса лекций, а также справочные материалы из документации на систему, полученные с помощью системной утилиты Linux — **man**.

Вся доступная при использовании утилиты **man** информация имеет многоуровневую структуру. Если документ имеет несколько таких уровней, то это отмечается в конце описания документа. Для того чтобы получить соответствующую информацию по команде операционной системы на первом уровне необходимо набрать строку **man [<уровень>] <имя команды>**. Номер первого уровня задается по умолчанию.

При вызове утилиты информация выдается на экран постранично. Для перемещений по тексту можно использовать стрелки вниз и вверх. Листание по страницам вперед выполняется по клавишам `f` или пробел. Листание назад по клавише **b**. Выход их просмотра документа выполняется по клавише **q**.

Утилита позволяет организовать поиск внутри текста руководства. Для того чтобы найти слово с просмотром текста вперед необходимо ввести **/<слово>**, для поиска назад - **?<слово>**. Продолжение поиска по заданному образцу производится по клавише **n**.

Для получения протоколов результатов выполнения работ следует перенаправлять данные из стандартного потока вывода в файл.

Требования к оформлению работ

По каждой лабораторной работе составляется отчет, который должен содержать:

- титульный лист;
- название и цель работы;
- лабораторное задание;
- для программы - описание данных и при необходимости описание структуры программы;

— текст программы или скрипта;
— результаты выполнения команд, скриптов и программ;
— выводы по результатам выполнения работы.

Отчет может представляться в виде твердой копии или в виде файла в формате редакторов LibreOffice или MS Word.

Лабораторная работа № 1. Команды Windows

Цель работы: ознакомиться с основными командами командной строки Windows

Работа с терминалом

Запуск терминала можеть выполнятся следующими способами.

Комбинация клавиш **Win+R** в открытом окне набирается команда **cmd.**

Кнопка **Пуск,** пункт **Служебные Windows,** пункт **Командная строка.**

Общий формат команды **<команда> [<ключи>] [<параметры>]**.

В открывшемся окне терминала можно получить справку по командам системы командой **HELP**. Для получения справки по конкретной команде используется строка: **HELP <имя команды>.**

Аналогично справку по отдельной команде можно просмотреть с использованием для команды ключей **/?**.

Команды системного назначения

CLS – Очистка экрана.

ECHO – Вывод сообщений и переключение режима отображения команд на экране.

DATE – Вывод либо установка текущей даты.

TIME – Вывод и установка системного времени.

SYSTEMINFO – Вывод сведений о системе и конфигурации компьютера.

PATH – Отображает или устанавливает путь поиска исполняемых файлов.

EXIT – Завершение работы интерпретатора командных строк.

Команды работы с каталогами

MD (**MKDIR**) – Создание каталога.

RD (**RMDIR**) – Удаление каталога.

CD (CHDIR) – Вывод имени либо смена текущего каталога.

DIR – Вывод списка файлов и подкаталогов из указанного каталога.

TREE – Графическое отображение структуры каталогов диска или каталога.

MOVE – Перемещение или переименование каталога.

XCOPY – Копирование структур каталогов.

Команды работы с файлами

COPY – Копирование одного или нескольких файлов в другое место. Команда может использоваться для создания текстового файла. В этом случае команда должна иметь вид **COPY CON <имя файла>**. Далее с клавиатуры вводится текст, которычй должен заканчиваться признаком конца файла **CTRL+Z** или клавишей **F6**.

TYPE – Вывод на экран содержимого текстовых файлов.

MORE – Последовательный вывод данных по частям размером в один экран.

DEL – Удаление одного или нескольких файлов.

MOVE – Перемещение одного или нескольких файлов из одного каталога в другой.

RENAME (REN) – Переименование файлов или каталогов**.**

FC – Сравнение файлов или наборов файлов и вывод различий между ними.

COMP – Сравнение содержимого двух файлов или двух наборов файлов.

MKLINK – Создание символических и жестких ссылок

FIND – Поиск текстовой строки в одном или нескольких файлах.

Перенаправление стандартных потоков ввода-вывода и ошибок

С помощью переназначения устройств ввода-вывода одна программа может направить свой вывода на вход другой или перехватить вывод другой программы, используя его в качестве своих входных данных. Для программ, которые используют стандартные входные и выходные устройства операционная система позволяет:

- выводить сообщения программ на экран (стандартный выходной поток), а в файл или на принтер (перенаправление вывода);
- читать входные данные не с клавиатуры (стандартный входной поток), а из заранее подготовленного файла (перенаправление ввода);
- передавать сообщения, выводимые одной программой, в качестве входных данных для другой программы (конвейер или композиция команд.

Для того, чтобы перенаправить текстовые сообщения, выводимые какой-либо командой из командной строки, в текстовой файл нужно использовать конструкцию

команда > имя_файла.

Если при этом заданный для вывода файл уже существовал, то он перезаписывается (старое содержимое теряется), если не существовал создается. Можно также не создавать файл заново, а дописывать информацию, выводимую командой, в конец существующего файла. Для этого команда перенаправления вывода должна быть задана так

команда >> имя_файла.

С помощью символа <, можно прочитать входные данные для заданной команды не с клавиатуры, а из заранее подготовленного файла

. **команда < имя_файла**.

В случае необходимости зафиксировать сообщения об ошибках стандартный поток ошибок можно перенаправить в текстовый файл с помощью команды

2 > имя_файла.

В этом случае стандартный поток ошибок (2) будет перенаправлен в файл.

Лабораторные задания

1. Просмотреть текущую дату и текущее время.
2. Создать каталог.
3. Создать в этом каталоге простой текстовый файл.
4. Вывести на экран файл.
5. Скопировать данный файл с другим именем.
6. Просмотреть содержимое каталога.
7. Скопировать данный каталог с другим именем.
8. Создать новый каталог.
9. Скопировать в него файл.
10. Сравнить файлы в каталогах.

Лабораторная работа № 2. Командные файлы Windows

Цель работы: научиться работать с командными файлами.

Методические указания

Основные команды для работы с командными файлами

CALL – Вызов одного пакетного файла из другого.

CMD – Запуск еще одного интерпретатора командных строк Windows.

ERASE – Удаление одного или нескольких файлов.

FOR – Запуск указанной команды для каждого из файлов в наборе.

GOTO – Передача управления в отмеченную строку пакетного файла.

HELP – Выводит справочную информацию о командах Windows.

IF – Оператор условного выполнения команд в пакетном файле.

PAUSE – Приостанавливает выполнение пакетного файла и выводит сообщение.

SET – Показывает, устанавливает и удаляет переменные среды Windows.

SORT – Сортировка ввода**.**

TASKLIST – Отображение всех выполняемых задач, включая службы.

TASKKILL – Прекращение или остановка процесса или приложения.

EXIT – Завершение работы интерпретатора командных строк.

Создание командного файла

Создать файл в текущем каталоге ***bat1.bat***

```
copy con mytext
echo mytext
```

Запустить терминал.

Выполнить командный файл.

После запуска набрать текст, завершив его <ctrl+Z> <Enter> или <F6> <Enter>.

Перенаправление вывода:

```
ECHO qwerty > mytext
```

Другой способ ввода значения переменной по команде set /P name=VALUE=Enter Value: позволяет ввести с клавиатуры строку и присвоить ее переменной name. Для того чтобы применить значение переменной ее необходимо записать %name%.

С помощью команды ECHO OFF можно отключить дублирование команд, идущих после нее (сама команда ECHO OFF при этом все же дублируется).

Файл ***bat2.cmd***

```
echo off
copy con mytext
echo mytext
```

Восстанавливается режим отображения командой echo on

Кроме этого, можно отключить дублирование любой отдельной строки в командном файле, написав в начале этой строки символ @.

```
@echo off
copy con mytext
echo mytext
```

Входные параметры для командного файла

Существует возможность передать командному файлу параметры командной строки и использовать их значения в операторах самого командного файла.

```
BAT-файл <параметр1>,<параметр2>,...<параметрN>
```

В самом командном файле первый параметр будет доступен как переменная %1, второй - %2 и т.п. Имя самого командного файла доступно как переменная %0.

Операторы перехода

Командный файл может содержать метки и команды GOTO перехода к этим меткам. Любая строка, начинающаяся с двоеточия, воспринимается при обработке командного файла как метка. Имя метки задается набором символов, следующих за двоеточием до первого пробела или конца строки.

Метка может использоваться в операторах перехода GO TO или CALL.

В команде перехода GOTO можно задавать в качестве метки перехода строку :EOF, которая передает управление в конец текущего пакетного файла.

Операторы условия.

С помощью команды IF ... ELSE (ключевое слово ELSE может отсутствовать) можно выполнять обработку условий.

Если заданное после IF условие принимает истинное значение, система выполняет следующую за условием команду (или несколько команд, заключенных в круглые скобки), в противном случае выполняется команда (или несколько команд в скобках), следующие за ключевым словом ELSE.

Проверка значения переменной.

```
IF [NOT] строка1==строка2 команда1 [ELSE команда2]
IF [/I] [NOT] строка1 оператор_сравнения строка2 команда
```

Строки могут быть литеральными или представлять собой значения переменных (например, ***%1*** или ***%TEMP%***). Кавычки для литеральных строк не требуются.

Синтаксис и значение операторов_сравнения представлены в таблице.

Оператор	Значение
EQL	Равно
NEQ	Не равно
LSS	Меньше
LEQ	Меньше или равно
GTR	Больше
GEQ	Больше или равно

Проверка существования заданного файла

```
IF [NOT] EXIST файл команда1 [ELSE команда2]
```

Условие считается истинным, если указанный файл существует. Кавычки для имени файла не требуются.

Цикл FOR … IN … DO …

```
FOR %%переменная IN (множество)
 DO команда [параметры]
```

Параметр множество в команде `FOR` задает одну или более текстовых строк, разделенных запятыми, которые вы хотите обработать с помощью заданной команды. Скобки здесь обязательны.

Параметр `команда [параметры]` задает команду, выполняемую для каждого элемента множества, при этом вложенность команд `FOR` на одной строке не допускается.

Параметр `%%переменная` представляет подставляемую переменную, причем здесь могут использоваться только имена переменных, состоящие из одной буквы.

При выполнении команда FOR заменяет подставляемую переменную текстом каждой строки в заданном множестве, пока команда, стоящая после ключевого слова DO, не обработает все такие строки.

Цикл FOR /D … IN … DO …

```
FOR /D %%переменная IN (набор) DO команда [параметры]
```

В случае, если набор содержит подстановочные знаки, то команда выполняется для всех подходящих имен каталогов, а не имен файлов.

Цикл FOR /R … IN … DO …

```
FOR /R [[диск:]путь] %%переменная IN (набор)
  DO команда [параметры]
```

В этом случае заданная команда выполняется для каталога [диск:]путь, а также для всех подкаталогов этого пути. Если после ключа R не указано имя каталога, то выполнение команды начинается с текущего каталога.

Цикл FOR /L … IN … DO …

Ключ /L позволяет реализовать с помощью команды FOR арифметический цикл, в этом случае синтаксис имеет следующий: вид:
`FOR /L %%переменная IN (начало,шаг,конец) DO команда [параметры]`

Здесь заданная после ключевого слова IN тройка (начало, шаг, конец) раскрывается в последовательность чисел с заданными началом, концом и шагом приращения.

Цикл FOR /F … IN … DO …

```
FOR /F ["ключи"] %%переменная IN (набор)
 DO команда [параметры]
```

Параметр набор содержит имена одного или нескольких файлов, которые по очереди открываются, читаются и обрабатываются.

Обработка состоит в чтении файла, разбиении его на отдельные строки текста и выделении из каждой строки заданного числа подстрок. Затем найденная подстрока используется в качестве значения переменной при выполнении основного тела цикла (заданной команды).

Ключи представляют собой заключенную в кавычки строку, содержащую ключевые слова.

Ключи в команде `FOR /F`

Ключ	Описание
EOL=C	Определение символа комментариев в начале строки (допускается задание только одного символа)
SKIP=N	Число пропускаемых при обработке строк в начале файла
DELIMS=XXX	Определение набора разделителей для замены, заданных по умолчанию пробела и знака табуляции
TOKENS=X,Y,M-N	Определение номеров подстрок, выделяемых из каждой строки файла и передаваемых для выполнения в тело цикла

Лабораторные задания

1. В соответствии с вариантом задания подготовить и выполнить сценарий, использующий условие.

1.1. Проверить существует ли в текущем каталоге файл и вывести на экран его содержимое, имя файла определяется через параметр.

1.2. Вывести на экран содержимое файла. Если файла в текущем каталоге нет, скопировать файл из другого каталога. Имя файла передается через параметр.

1.3. Прочитать содержимое указанного каталога в файл. Если каталог пуст, выдать на экран сообщение. Имя каталога вводится с клавиатуры.

1.4. Просмотреть содержимое текущего каталога, ввести имя одного из файлов. Если этот файл существует, то вывести его содержимое на экран.

1.5. Если указанный в параметре файл не имеет установленного атрибута разрешения записи, то необходимо установить этот параметр.

1.6. Проверить существует ли в текущем каталоге файл, имя файла вводится с клавиатуры. Если файл не существует, сформировать его.

1.7. Прочитать содержимое указанного каталога в файл. Если каталог пуст, выдать на экран сообщение. Имя каталога передается через параметр.

1.8. Вывести на экран содержимое файла. Если файла в текущем каталоге нет, скопировать файл из другого каталога. Имя файла вводится с клавиатуры. Имя каталога передается через параметр.

1.9. Проверить является ли указанный в параметре файл каталогом. Вывести соответствующую информацию на экран. Если это каталог, то установить разрешение записи в этот каталог.

2. Перед выполнением задания создать в своем каталоге набор файлов с расширениями *.c, *.cpp, *.h и *.txt.

2.1. Используя цикл ***for***, распечатать из текущего каталога содержимое всех файлов с расширениями ***.c** и ***.cpp**. Перед содержимым каждого файла напечатать его имя.

2.2. Используя цикл for, создать в каталоге ***"./links"*** символические ссылки на все файлы текущего каталога с добавлением к имени файла ***".link"***.

2.3. Синхронизировать содержимое каталогов ***"./"*** и **"./backup"** путем создания символических ссылок на недостающие файлы.

2.4. Вывести для определенных каталогов имена текстовых файлов, для которых разрешена запись. Имена каталогов задаются через параметры.

2.5. Проверить существует ли в каталогах, заданных через параметры при вызове сценарии, файл. Имя файла вводится с клавиатуры.

2.6. Вывести для каталога (имя каталога вводится с клавиатуры) список файлов, для которых разрешены чтение.

2.7. Создать резервные копии текстовых файлов, имеющих атрибут разрешения для записи.

2.8. Проверить установлены ли атрибут разрешения чтения для файлов, имена которых перечисляются в списке параметров при вызове сценария.

2.9. Преобразовать имена файлов с расширением ***h*** в имена с расширением ***hpp***. Каталоги, в которых выполняются преобразования, задаются через параметры.

Лабораторная работа № 3. Команды Shell операционной системы Linux

Цель работы: изучить особенности формирования и выполнения некоторых команд оболочки Bourn shell операционной системы Linux.

Методические указания

Интерфейс операционной системы представлен командным интерпретатором shell. В лабораторном практикуме будет использоваться командный интерпретатор Bourn again shell (bash). Выполнение команд должно производиться в терминале. Для работы с текстовыми файлами можно использовать любой текстовый редактор, например gedit или kate.

В общем случае команда имеет следующий вид:

<команда> <ключи> <аргументы>

Ключ определяет режим выполнения команды и представляет собой символ, перед которым указывается знак **-**, например, -d. Возможно объединение нескольких ключей в группу: -al.

Команда просмотра содержимого каталога ls выдает список имен файлов и каталогов. Общий вид команды

ls [<ключи>] [<аргументы>]

Команда cat выводит содержимое файла на экран терминала. Команда more выводит содержимое файла поэкранно. Для управления просмотром так же как при работе с утилитой man можно использовать клавиши f, b и q.

Команда pwd выдает на экран абсолютное путевое имя текущего каталога. Команда cd изменяет текущий каталог. При использовании команды без параметра выполняется переход в домашний каталог пользователя.

Команда копирования cp файлов имеет формат:

cp [ключи] <исходный файл> <результирующий файл>

Команда перемещения (переименования) имеет формат:

mv [<ключи>] <текущее имя файла> <новое имя файла>,

для перемещения файла в другой каталог используется команда:

mv [<ключи>] <имя файла> <имя каталога>

В файловой системе ОС Linux один файл может иметь несколько имен. Этот механизм основан на том, что с помощью специальной команды ln могут создаваться специальные структуры данных - связи. Связи бывают жесткими и символическими. В первом случае связь хранится в управляющих метаданных файла. Во втором - в каталоге создается специальный файл, содержащий имя целевого файла.

Команда создания связи **ln**. Формат команды:

ln [<ключи>] <имя файла> <имя файла>

Команда ln с ключом -s создает символическую связь.

Команда удаления файла rm

rm [<ключи>] <имя файла>

Команда удаляет имя файла из каталога, при этом уменьшается на 1 счетчик жестких ссылок файла. Если счетчик становиться равным 0, то файл уничтожается.

При использовании ключа `-i` запрашивается подтверждение на выполнение операции. Команда `rm` может использоваться и для удаления каталогов, для этого она должна использоваться с ключом `-r`. При этом сначала удаляется все содержимое каталога, а затем и сам каталог. Для удаления пустых каталогов служит команда `rmdir`.

Логически все файлы ОС Linux организованы как последовательность байтов. Эта организация распространяется и на операции ввода и вывода. Все вводимые данные представляют собой поток байтов, называемый входным потоком (стандартный ввод). Выводимая информация организована также как последовательный поток данных, направляемых на терминал - стандартный вывод.

Стандартные ввод и вывод могут взаимодействовать с файлами с помощью операции переназначения (переадресации). При этом результаты выполнения любой команды можно передать в файл. Например, после выполнения команды

```
ls -l > dir_file
```

будет сформирован файл с именем `dir_file`. Этот файл будет содержать полный список файлов, находящихся в текущем каталоге.

С помощью переназначения выхода можно сформировать файл, содержащий произвольную строку. Для этого используется команда `echo`, которая переадресует свой выход в файл:

```
echo my string 1 > my_file
```

При попытке переназначить следующую строку в тот же файл, старое содержимое файла будет уничтожено. Чтобы этого не произошло необходимо при переназначении использовать символы "**>>**". Так после выполнения команды

```
echo my string 2 >> my_file
```

в файл будет добавлена новая строка.

Если ввести команду `cat` без параметров, то по умолчанию входным файлом для нее будет считаться стандартный вход, а выходом —

стандартный выход. Можно переназначить выход команды в файл на диске `cat > file1`, таким образом можно получить любой текстовый файл. Для завершения ввода необходимо ввести символ конца файла — `Ctrl+d`.

Аналогично могут быть перенаправлены и входные данные. Для обозначения перенаправления входа используется знак **<**. При этом следует учитывать, что вход данных из файла может использоваться только для тех команд, которые получают данные от стандартного входа. Например, следующая команда обеспечивает подсчет количества строк в файле.

```
wc -l < file1
```

Операции переназначения можно объединять. Так при выполнении команды

```
cat < file1 > file2
```

будет произведено копирование содержимого файла `file1` в файл `file2`.

В отличии от переназначения входов и выходов программные каналы (конвейеры) имеют другую природу. Они являются простым способом организации взаимодействия процессов. Поэтому такой канал перенаправляет данные с выхода одного процесса (в нашем случае команды) на вход другого процесса (команды). Такое перенаправление обозначается символом "**|**". В общем случае конструкция может быть иметь вид

```
<команда> <параметры> | <команда> <параметры>
```

Так команда

```
cat -n my_file | lpr
```

выводит содержимое файла `my_file` на печать. Текст печатается с номерами строк.

Использование программных каналов существенно расширяет возможности команды `more` и фильтров `sort` и `grep`.

Фильтр `grep` ищет в файлах образец и перечисляет все строки, в которых он есть. Фильтр `sort` выводит отсортированную версию файла. При этом сам файл не изменяется.

Порядок выполнения работы

1. Командой `echo -n > имя файла` создать файл. С помощью команд `cat` и `ls` просмотреть его содержимое и длину.
2. Задействовав ряд ключей команды `ls`, получить результат. Объяснить значение каждого поля.
3. Открыть файл в текстовом редакторе. Набрать несколько осмысленных строк. Завершить работу с редактором.
4. Просмотреть содержимое файла, используя команду `cat`. Повторно войти в редактор и изменить файл таким образом, чтобы количество строк в нем превышало 25 (число строк на экране). Вновь вывести файл на экран. Объяснить полученные результаты.
5. Используя команду `more`, добиться поэкранного вывода файла. Объяснить полученные результаты.
6. Командой `mkdir <имя каталога>` создать каталог. Скопировать туда созданный файл.
7. Попытаться произвести повторное копирование. Объяснить полученные результаты. Используя опции команды `cp`, добиться результата.
8. Скопировать в каталог тот же файл, указав в качестве приемника имя, отличающееся от старого лишь последним символом. Повторить операцию 4 раза.
9. Командой `cd <имя каталога>` перейти в созданный каталог. Скопировать в домашний каталог 4 файла, используя разные виды масок (`*`, `?`, `[]`). Необходимо каждый раз просматривать содержимое каталога командой `ls -l`. После каждого копирования удалять файлы в каталоге - адресате командой `rm`, используя необходимые ключи.
10. Удалить все созданные файлы, используя команду `rm <имя файла>`. Во избежание удаления файлов, созданных другой бригадой, при выполнении этого пункта задания, запрещается пользоваться маской "`*`".

11. Создать в рабочем каталоге подкаталог.
12. В соответствии с лабораторным заданием создать пример текстового файла. Файл должен содержать несколько строк информации.
13. В соответствии с вариантом задания создать файл с содержимым каталога. Просмотреть полученный файл.
14. Используя программный канал, обработать результаты выполнения заданной команды соответствующим фильтром. Результат сохранить в файле. команда `who` выводит список работающих в текущий момент пользователей, команда `head` выводит первые строки файла. По умолчанию выводятся 10 первых строк. Это значение можно изменить, используя ключ вида `-<число строк>`.

Варианты заданий

№	Пункт 13 Операция	Пункт 14 команда	фильтр
1	Сокращенный формат содержимого текущего каталога	`ls`	`grep`
2	Полная информация по файлам	`ls`	`sort`
3	Содержимое каталога рекурсивно вместе с подкаталогами	`ls`	`more`
4	Вывод всех элементов каталога без исключения	`cat`	`grep`
5	С указанием размера файла в блоках	`cat`	`sort`
6	С добавлением признака каталога (/) после его имени	`cat`	`more`
7	Сортировать по размеру блока в обратном порядке	`who`	`grep`
8	С выводом элементов каталога в несколько столбцов	`who`	`sort`
9	С сортировкой по времени последней модификации файла	`who`	`more`

№	Пункт 13 Операция	Пункт 14 команда фильтр
10	Сортировать по времени создания файла	`head grep`
11	Сортировать по времени создания файла (в обратном порядке)	`head sort`
12	С сортировкой по времени последней модификации файла (в обратном порядке)	`head more`

Лабораторная работа № 4. Работа с файлами

Цель работы: изучение особенностей некоторых команд для работы с файлами.

Методические указания

При выполнении работы необходимо использовать следующие команды.

`file [-f<файл>] <имена файлов>`

Команда классификации файлов. При выполнении этой команды система по ключевой информации, имеющейся в файле или по содержимому файла пытается определить тип этого файла. Для текстового файла делается попытка распознать язык программирования. При использовании ключа `-f` имена файлов извлекаются из файла имя, которого определяется непосредственно за ключом.

`find <путь> <выражение>`

Поиск файла. Первый параметр задает список каталогов, в которых ведется поиск файлов, второй - критерии поиска. В результате выполнения команды выводятся имена файлов для которых значение выражения истинно. Ключи команды можно просмотреть пои вызове страницы справочника `man find`.

od [<ключ>] <файл>

Вывод дампа файла. Формат вывода определяется ключом: восьмеричный (**-o**), шестнадцатеричный (**-x**), десятичный (**-d**) или символьный (**-c**). Если используется ключ **-b**, то дамп выводится побайтно в восьмеричной форме.

cmp [-l] [-s] <файл1> <файл2>

Команда сравнивает файл1 и файл2. Если эти файлы различаются, то выводятся несовпадающие строки. Команду обычно используют для сравнения бинарных файлов.

-l вывод десятичного номера не совпадающего байта и его восьмеричного значения.

-s возврат только кода завершения: 0 - файлы совпадают, 1 - файлы имеют различия, 2 - сравнение невозможно (файл не найден).

diff [<ключ>] <файл1> <файл2>

Команда выводит информацию о несовпадениях файлов. Вывод производится в формате строкового редактора ed.

Файлы в Linux имеют двух владельцев: пользователя и группу. Владельцем-пользователем вновь созданного файла является пользователь, создавший этот файл. Этот пользователь может изменить права доступа к нему по специальной команде.

Права доступа могут изменяться с использованием команды

chmod <режим> <список файлов>

Эта команда может задавать правила изменения атрибутов доступа (<режим>) в двух формах - символической и числовой. В символической форме атрибуты задаются в формате

[who] + | - = [permission]

Первое поле who может содержать один из следующих символов:

a установка атрибутов защиты для всех категорий пользователей (используется по умолчанию);

g установка атрибутов защиты для групп пользователей;

o установка для прочих процессов;

u установка атрибутов только для владельца.

Допустимые операции:

+ добавление прав доступа;

− отмена прав доступа;

= установка перечисленных и отмена остальных прав.

Параметр `permission` устанавливает значения атрибутов для перечисленных в первом параметре процессов. Этот параметр может представляться комбинацией из следующих символов:

`x` - разрешает выполнение (для каталога разрешение поиска);

`r` - разрешение чтения;

`w` - разрешение записи;

`s` - установка пользовательского или группового идентификатора.

Команда `chmod g+x-w exampl` устанавливает для группы право на выполнение файла `exampl` и снимет право на запись.

При использовании числовой формы определения атрибутов они задаются в виде восьмеричных кодов, каждая восьмеричная цифра задает атрибуты соответственно для владельца, группы и остальных процессов. Двоичные разряды восьмеричной цифры определяют разрешение - 1 или запрет - 0 последовательно для чтения, записи и исполнения. Например, команда

```
chmod 754 my_file
```

дает все права владельцу, разрешает чтение и исполнение для группы и устанавливает право на чтение для всех остальных пользователей.

Исполняемый файл может быть откомпилированной программой или командным файлом интерпретатора shell. В последнем случае должно быть установлено право на чтение, т.к. командный интерпретатор должен иметь возможность считывать команды из файла.

Права доступа для каталогов имеют свои особенности. Право чтения из каталога дает возможность получить только имена находящихся в каталоге файлов. Для получения дополнительной информации о файлах необходимо иметь разрешение на работу с метаданными файла и использованием атрибута исполнение. Право на исполнение необходимо также для того, чтобы сделать каталог текущим. Используя эти свойства, можно сделать содержимое каталога невидимым для просмотра, но

отдельные файлы из такого каталога будут доступны. Для этого необходимо запретить чтение из каталога, но установить для него атрибут исполнения.

Лабораторные задания

1. Создать текстовый файл содержащий не менее 5 строк текста.
2. Создать несколько ссылок, просмотреть результаты по команде `ls -l`. Просмотреть файл, обратившись к нему по ссылке. Удалить одну ссылку и снова просмотреть результаты.
3. Создать символическую ссылку и просмотреть результат.
4. Определить тип созданного файла с использованием команды `file`.
5. Просмотреть дамп файла. Формат просмотра задается в соответствии с вариантом задания: побайтно — варианты 1, 6, 11; коды символов - 2, 7, 12; десятичные числа - 3, 8; восьмеричные числа - 4, 9; шестнадцатеричные числа - 5, 10.
6. Скопировать файл в тот же каталог с другим именем. Используя редактор изменить две строки файла-копии.
7. Выполнить сравнение файлов используя команду `cmp` - варианты 1, 3, 5, 7, 9, 11; команду `diff` - варианты 2, 4, 6, 8, 10, 12.

Выполнить поиск файлов по следующим правилам. По числу связей - варианты 1, 7; по имени владельца - варианты 2, 8; по имени группы - варианты 3, 9; по размеру в байтах - варианты 4, 10; по размеру в блоках - варианты 5, 11; по времени модификации - варианты 6, 12.

8. В рабочем каталоге создать файл и проанализировать его атрибуты.
9. Изменить атрибуты каталога и проанализировать результат. Изменение выполнять в соответствии с вариантом.

Чтение для владельца - 1, 4, 7, 10.

Запись для владельца -2, 5, 8, 11.

Чтение и запись для владельца - 3, 6, 9, 12.

10. Создать каталог и скопировать в него файл. Проанализировать атрибуты этого каталога.
11. Изменить атрибуты каталога и проанализировать результат.

Чтение для владельца - 1, 6, 11. Запись для владельца - 2, 7. Чтение и запись для владельца - 3, 8. Исполнение для владельца - 4, 9, 12. Чтение и исполнение для владельца - 5, 10.

Лабораторная работа № 5. Управление процессами

Цель работы: ознакомится с некоторыми командами для управления процессами в операционной системе Linux.

Методические указания

Каждый процесс имеет уникальный идентификатор (PID). Этот идентификатор позволяет системе различать процессы.

Определение значений идентификаторов может быть выполнено с использованием команды

```
ps [<ключи>] [<терминал>] [<сортировка>] [<ID>]
```

При использовании различных ключей эта команда позволяет получить информацию о процессе.

Примерами ключей команды могут служить:

`l` длинный формат;

`a` показывать процессы других пользователей;

`e` показывать процессы всех пользователей;

`r` показывать только выполняющиеся процессы.

Полностью значения ключей можно просмотреть с соответствующей странице справочника (следует учитывать, что в некоторых версиях операционной системы значения ключей могут не совпадать).

Параметр txx определяет вывод информации только для процессов, связанных с указанным терминалом. Параметр сортировка задает порядок вывода информации. Указав последний параметр можно вывести данные только для определенных параметром процессов.

Команда

```
who [<ключи>]
```

выдает список пользователей. Ключи этой команды могут иметь следующие значения:

q выводятся только имена и идентификаторы пользователей;

i, u после имени выводится время;

H выводятся заголовки.

Команда id выводит идентификатор пользователя и группы для текущего процесса.

Одним из способов передачи информации между процессами служат сигналы. Из среды интерпретатора можно отправлять сигналы процессам с помощью команды

```
kill [-sig] pid
```

посылает процессу сигнал, заданный аргументом sig. В качестве второго аргумента используется идентификатор процесса, который может быть получен по команде ps. Команда kill -l возвращает список символических имен сигналов.

Лабораторные задания

1. Выполнить команду ps с различными ключами. Проанализировать результаты. Отразить результаты в отчете по работе.
2. Выполнить команду who с различными ключами. Проанализировать результаты. Отразить результаты в отчете по работе.
3. Открыть окно второго терминала. Запустить любую программу. Возвратиться на первый терминал и завершить работу программы на втором терминале по команде

`kill -9 <идентификатор процесса>`. Проверить результат завершения на втором терминале и по команде `ps`.

Лабораторная работа № 6. Командные файлы

Цель работы: научиться создавать командные файлы с использованием переменных окружения и команды ветвлений.

Методические указания

Стандартный интерпретатор имеет развитый язык программирования, позволяющий создавать командные файлы (скрипты). Скрипт представляет собой обычный текстовый файл, который включает в себя команды, функции и комментарии. Комментарии начинаются с символа "#". Для идентификации типа интерпретатора целесообразно в первой строке файла указывать соответствующий тип, например, `#!/bin/bash`. Командный файл обязательно должен иметь атрибут исполнимый.

Как и в любом языке программирования в командном интерпретаторе существуют переменные. Переменные обозначаются с помощью имен.

Значения присваиваются переменным по оператору присваивания, например, `length=80`.

Значение переменной можно получить, если перед ее именем указать знак **$**. Так при выполнении команды

```
echo $length
```

на экран будет выведено `80`. При работе с переменными интерпретатора следует помнить, что значением переменной всегда является строка.

Если переменной присваивается строковое значение, то можно использовать и кавычки и апострофы. Если в строке имеются пробелы, кавычки обязательны, поскольку они позволяют трактовать все символы, включая служебные как элементы строки, а не как специальные

управляющие символы. Исключением является символ **$** перед именем переменной, который всегда приводит к ее вычислению. В случае использования апострофа знак доллара не приводит к вычислению значения переменной.

Значения всех установленных переменных можно просмотреть по команде `set`.

Особое значение имеют обратные апострофы. Они позволяют получать значения переменных как результаты выполнения команд. Если некоторая команда будет заключена в обратные апострофы, то после ее исполнения результаты будут помещены на то место, где эта команда находится.

Важную роль для организации сценариев играют встроенные переменные:

`$1,$2,...` - параметры, задаваемые при запуске сценария;

`$#` - число позиционных параметров;

`$?` - код возврата последнего выполненного процесса;

`$$` - идентификатор текущего shell;

`$!` - идентификатор последнего процесса в фоновом режиме;

`$*` - все параметры, переданные сценарию (как единая строка);

`$@` - параметры передаются как слова, заключенные в кавычки.

Ветвления в сценарии организуются по команде

```
if <команда>
then
<команды>
else
<команды>
fi
```

После ключевого слова `if` используется специальная команда проверки условий `test`.

Команда используется в двух формах.

```
test <значение> -<опция> <значение>
```

или

```
test <строка> = <строка>
```

Опции задают соответствующие условия. Для различных типов операндов используются разные опции.

Опции для целых операндов:

`-gt` - больше;

`-lt` - меньше;

`-ge` - больше или равно;

`-le` - меньше или равно;

`-eq` - равно;

`-ne` - не равно.

Опции для строк:

`-z` - строка нулевой длины;

`-n` - проверка на строковое значение;

`=` - проверка на равенство строк;

`!=` - проверка на неравенство строк;

`-str` - проверка на строку ненулевой длины.

При проверке условий можно использовать сложные логические выражения с использованием логических операций. Для представления логических операций используются следующие опции:

`-a` - логическое И;

`-o` - логическое ИЛИ;

`!` - отрицание.

Для работы с файлами существуют специальные опции:

`-f` - файл существует и является обычным;

`-s` - файл не пустой;

`-r` - файл доступен для чтения;

`-w` - файл доступен для записи;

`-x` - файл доступен для исполнения;

`-d` - каталог;

`-h` - символьная ссылка.

Вместо ключевого слова `test` в команде проверки условия можно использовать квадратные скобки, которые отделяются от условия

пробелом, табуляцией или символом перехода на новую строку. Два следующих выражения будут эквивалентными:

```
test $num -eq 10
[ $num -eq 10 ]
```

Оператор множественного выбора **case** имеет следующий синтаксис:

```
case строка in
     шаблон) список команд;;
     шаблон) список команд;;
...
esac
```

Данный оператор в основном тождественен широко применяемой в языке C конструкции `switch`. Однако в отличии от последней здесь не требуется явным образом использовать оператор `break` - в случае совпадения одного из условий с образцом дальнейший перебор вариантов прекращается автоматически.

Если необходимо определить действие по умолчанию, то есть на тот случай, если строка не совпала ни с одним из шаблонов, то в поле образца необходимо поставить символ "*****".

Одним из наиболее мощных средств автоматизации выполнения операций в shell является оператор цикла с перечислением **for**, который в общем виде можно записать следующим образом:

```
for переменная [in список_значений]
     do
          список команд
     done
```

Здесь переменная последовательно принимает значения из списка. Если список значений переменной отсутствует, то при организации цикла используются параметры командной строки при вызове скрипта.

Особого упоминания заслуживает конструкция `for <переменная> in *` которая обозначает "все файлы и подкаталоги внутри данного каталога". Она широко используется, например, при анализе содержимого требуемого каталога.

Лабораторные задания

1. Подготовить и выполнить сценарий, содержащий установку и отображение значений переменных.

2. В соответствии с вариантом задания подготовить и выполнить сценарий, использующий условие.

2.1. Проверить существует ли в текущем каталоге файл и вывести на экран его содержимое, имя файла определяется через параметр. Если файл не существует, сформировать файл с помощью команды `cat`.

2.2. Определить количество работающих в данный момент пользователей. Если число пользователей больше 10 выдать сообщение.

2.3. Вывести на экран содержимое файла. Если файла в текущем каталоге нет, скопировать файл из другого каталога. Имя файла передается через параметр.

2.4. Прочитать содержимое указанного каталога в файл. Если каталог пуст, выдать на экран сообщение. Имя каталога вводится с клавиатуры.

2.5. Просмотреть содержимое текущего каталога, ввести имя одного из файлов. Если этот файл имеет ненулевую длину, то вывести его содержимое на экран.

2.6. Если указанный в параметре файл не имеет установленного атрибута разрешения, то необходимо установить этот параметр.

2.7. Проверить существует ли в текущем каталоге файл, имя файла вводится с клавиатуры. Если файл не существует, сформировать его.

2.8. Определить количество файлов с расширением `txt`, находящихся в текущем каталоге. Если таких файлов нет, то выдать на экран сообщение.

2.9. Прочитать содержимое указанного каталога в файл. Если каталог не пуст, выдать на экран сообщение. Имя каталога передается через параметр.

2.10. Вывести на экран содержимое файла. Если файла в текущем каталоге нет, скопировать файл из другого каталога. Имя файла вводится с клавиатуры. Имя каталога передается через параметр.

2.11. Определить количество работающих в настоящее время пользователей с заданным именем. Если количество пользователей больше числа передаваемого в сценарий через параметр, вывести сообщение.

2.12. Проверить является ли указанный в параметре файл каталогом. Вывести соответствующую информацию на экран. Если это каталог, то установить разрешение записи в этот каталог.

3. Используя оператор множественного выбора `case`, разработать и отладить сценарий на языке shell, реализующий меню типа

Введите нужный пункт меню:

1. Операция 1

2. Операция 2

Имена файлов, образцы для фильтров должны передаваться в качестве параметров. Анализируемые в меню значения вводятся с клавиатуры.

Варианты заданий

Операция 1:	Произвести поиск всех файлов, которые изменялись столько дней назад, каков номер варианта. Поиск ведется от домашнего каталога. При этом не указывать полный путь. Имена файлов приемников результата и потока ошибок принимаются соответствующими номеру варианта с префиксами `out` и `err` соответственно. Обоим файлам присвоить расширение, соответствующее: (1,4,9,12) - номеру процесса, (2,6,8,11) - номеру последнего фонового процесса, (3,5,7,10) - номеру варианта.

Операция 2:	Произвести поиск всех процессов, принадлежащих (варианты 1,3,4,7,9,12) или нет (2,5,6,8,10,11) конкретному пользователю, зарегистрированному в системе. Имя пользователя, как и имя файла-приемника, вводится с клавиатуры. Вывод команды должен быть отсортирован в алфавитном порядке. У файла-приемника должно быть расширение, соответствующее введенному идентификатору пользователя.

4. Перед выполнением задания создать в своем каталоге набор файлов с расширениями `*.c, *.cpp, *.h и *.txt`.

4.1. Используя цикл `for`, распечатать из текущего каталога содержимое всех файлов с расширениями `*.c` и `*.cpp`, в именах которых присутствуют цифры. Перед содержимым каждого файла напечатать его имя.

4.2. Используя цикл `for`, создать в каталоге `"./links"` символические ссылки на все файлы текущего каталога с добавлением к имени файла `".link"`.

4.3. Синхронизировать содержимое каталогов `"./"` и `"./backup"` путем создания символических ссылок на недостающие файлы.

4.4. Вывести для определенных каталогов имена текстовых файлов, для которых разрешена запись. Имена каталогов задаются через параметры.

4.5. Проверить существует ли в каталогах, заданных через параметры при вызове сценарии, файл. Имя файла вводится с клавиатуры.

4.6. Проверить работают ли в настоящее время в системе пользователи, имена которых заданы списком параметров.

4.7. Вывести для каталога (имя каталога вводится с клавиатуры) список файлов, для которых разрешены исполнение и чтение.

4.8. Создать резервные копии текстовых файлов, имеющих атрибут разрешения для записи.

4.9. Копировать в каталог, имя которого вводится с клавиатуры, файлы, у которых имя начинается с букв "a" или "z", если эти файлы не являются каталогами.

4.10. Проверить имеются ли в каталогах, имена которых определяются параметрами сценария, файлы, являющиеся исходными текстами программ.

4.11. Проверить установлены ли атрибуты разрешения чтения и исполнения для файлов, имена которых перечисляются в списке параметров при вызове сценария.

4.12. Преобразовать имена файлов с расширением h в имена с расширением hpp. Каталоги, в которых выполняются преобразования, задаются через параметры.