МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
Федеральное государственное бюджетное
образовательное учреждение
высшего профессионального образования
ОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
им. Ф.М. ДОСТОЕВСКОГО

Д.М. Бречка

ОПЕРАЦИОННЫЕ СИСТЕМЫ.
ЧАСТЬ 1.
ПАКЕТНЫЕ ФАЙЛЫ И УПРАВЛЕНИЕ КОМПЬЮТЕРОМ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

ОМСК-2012

УДК 004.451
ББК 32.973.2
Б877

Рецензенты:
канд. тех. наук, доц. Д.Н. Лавров
канд. физ.-мат. наук, А.Н. Кобанов

Бречка, Д.М.
Б877 Операционные системы. Часть 1. Пакетные файлы и управление: учебно-методическое пособие / Д.М. Бречка. – Омск: Изд-во Ом. гос. ун-та, 2012 – 68с.

Рассматриваются вопросы управления операционными системами Windows и Linux с помощь встоенных инструментов.

Пособие имеет пратическую направленность в области основ работы с операционными системами семейства Windows и Unix, и управлению компьютером с помощью пакетных файлов.

Соответствует государственным образовательным стандартам высшего профессионального образования по специальностям 230101.65 «Вычислительнве машины, комплексы системы и сети», 090102.65 «Компьютерная безопасность» и направлениям подготовки бакалавриата 090900.62 «Информационная безопасность», 230100 «Информатика и вычислительная техника».

Предназначено для выполнения лабораторных работ по курсу «Операционные системы».

УДК 004.451
ББК 32.973.2

Рекомендовано ученым советом факультета компьютерных наук ОмГУ

1. Ознакомление с пакетными файлами

1.1. Теоретический блок

1.1.1. Операционная система DOS

DOS (*Disk Operating System* — дисковая операционная система, ДОС) — семейство операционных систем для персональных компьютеров.

DOS является однозадачной операционной системой. После запуска управление передаётся прикладной программе, которая получает в своё распоряжение все ресурсы компьютера и может осуществлять ввод/вывод посредством как функций предоставляемых операционной системой, так и функций базовой системы ввода/вывода (BIOS), а также работать с устройствами напрямую.

DOS имеет консольную систему ввода/вывода. К**омандная строка –** содержит приглашение DOS, отображает информацию о текущем диске и текущем каталоге.

Основные команды DOS

Приведем основные команды, необходимые для работы с DOS. В квадратных скобках **[…]** указываются не обязательные параметры команды, в треугольных **<…>** – обязательные.

имя_диска: — команда смены текущего диска.

cd [диск:] <путь> - изменение текущего каталога.

dir [диск:][путь\][имя-файла] [параметры] - просмотр каталога.

md [диск:] <путь> - создание каталога.

rd [диск:] <путь> - уничтожение каталога.

copy con <имя файла> - создание текстовых файлов. После ввода этой команды нужно будет поочередно вводить строки файла. В конце каждой строки нужно нажимать клавишу Enter, а после ввода последней – нажать клавишу Ctrl+Z или F6 и затем Enter.

del <имя файла> - удаление файлов.

ren <имя файла1> <имя файла2> - переименование файлов.

copy <имя файла1> <имя файла2> или **copy <имя-файла1> [имя-каталога]** - копирование файлов.

copy <имя файла1> [+ имя-файла2] <имя файла3> - соединение (конкатенация) файлов.

type <имя файла> - вывод файла на экран.

cls - очистка экрана монитора

vol - вывод метки диска

tree [диск:][путь] [/F] – выводит дерево каталогов. Параметр [путь] задает каталог, для которого вы хотите вывести структуру. Параметр /F выводит имена файлов в каждом каталоге.

color [цвета] – изменение цвета экрана и текста/ атрибуты цвета

задаются двумя шестнадцатеричными цифрами: первая устанавливает цвет экрана, вторая - цвет текста.

edit – вызов текстового редактора MS DOS.

ключ /? – выводит справку о команде – ***команда/?***

help – вызов справки.

Перенаправление вывода ***имя_команды>имя_файла.txt*** – передает результат выполнения команды ***имя_команды*** в файл ***имя_файла.txt*** (перенаправляет вывод из консоли в файл). Например, команда ***dir>dir.txt*** позволяет сохранить оглавление текущего каталога в виде текстового файла ***dir.txt.***

Пакетные файлы DOS

Пакетные (командные) файлы - средство MS-DOS, позволяющее автоматизировать часто выполняемые действия пользователя. Пакетные файлы могут выполнять довольно сложную последовательность действий. Основой пакетных файлов служат команды MS-DOS. Пакетный файл представляет собой обычный текстовый файл, расширение которого меняется на **.BAT** (batch file – пакетный файл). Для запуска командного файла достаточно набрать в командной строке его имя, расширение можно не указывать.

Командный файл состоит из последовательности строк, в каждой из которых, может находиться либо вызов программы, либо вспомогательные команды. 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:

ECHO

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.

GOTO

Применяется для перехода к определенной строке командного файла. При выполнении команды **GOTO** ***LABEL*** происходит переход к строке, начинающейся с текста **:*LABEL*.**

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**.

PAUSE

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

REM

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

Пример. Пакетный файл создает папку и копирует туда содержимое текущей папки. На рисунке 1.1 приведен вывод на экран разультатов работы данного пакетного файла

Текст файла
echo off
cls
echo start backup
mkdir backup
*copy *.* backup*
echo end backup

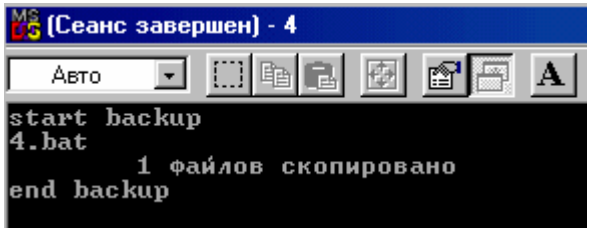

Рисунок 1.1 - Выполнение пакетного файла в командной строке

1.1.2. Операционная система UNIX

UNIX — группа переносимых, многозадачных и многопользовательских операционных систем.

Первая система UNIX была разработана в 1969 г. в подразделении Bell Labs компании AT&T. С тех пор было создано большое количество различных UNIX-систем. Юридически лишь некоторые из них имеют полное право называться «UNIX»; остальные же, хотя и используют сходные концепции и технологии, объединяются термином *«UNIX-подобные»*. Для краткости под UNIX-системами будем подразумевать как истинные UNIX, так и UNIX-подобные ОС.

Командный язык ОС Unix - shell оперирует с командами. Shell фактически есть язык программирования очень высокого уровня. На этом языке пользователь осуществляет управление компьютером. Обычно, после входа в систему вы начинаете взаимодействовать с командной оболочкой.

Признаком того, что оболочка (shell) готова к приему команд, служит выдаваемое ею на экран приглашение. В простейшем случае это один доллар ("$").

Shell не является необходимым и единственным командным языком (хотя именно он стандартизован в рамках POSIX [POSIX 1003.2] - стандарта мобильных систем). Например, немалой популярностью пользуется язык cshell, есть также kshell, bashell (из наиболее популярных в последнее время) и другие. Более того, каждый пользователь может создать свой командный язык. Может одновременно на одном экземпляре операционной системы работать с разными командными языками.

Shell - это одна из многих команд UNIX. То есть в набор команд оболочки (интерпретатора) "shell" входит команда "sh" - вызов интерпретатора "shell". Первый "shell" вызывается автоматически при вашем входе в систему и выдает на экран приглашение. После этого вы можете вызывать на выполнение любые команды, в том числе и снова сам "shell", который вам создаст новую оболочку внутри прежней.

Основные команды shell

Приведем основные команды shell. Следует помнить, что в UNIX различаются регистры символов, то есть имена ***File, file*** и ***FILE*** будут являться тремя разными именами файла.

cd <путь> - изменение текущего каталога.

ls [путь/][имя файла] [параметры] - просмотр каталога.

mkdir <путь> - создание каталога.

rmdir <путь> - уничтожение каталога.

cat> файл - создание пустого текстового файла.

rm <имя файла> - удаление файлов.

mv <имя файла1> <имя-файла2> - переименование файлов.

cp <имя файла1> … <имя файла N> <имя каталога> - копирование файлов.

rm -r <имя каталога> - удаление каталога со всем содержимым.

cat <имя файла1> …<имя файла N> - вывод файлов на экран.

clear - очистка экрана монитора

grep <шаблон> **<имя файла1> …<имя файла N> -** выдает все строки в названном файле (файлах), которые содержат заданный шаблон.

vi – вызов текстового редактора vi.

ключ –h (--help) – вывод справки о команде - ***команда -- help***

man – справочная информация - ***man команда***

Также как и для DOS можно применять перенаправление вывода, например ***ls>file1.txt*** выведет содержимое текущего каталога в файл ***file1.txt.***

Пакетные файлы shell

Также как и в DOS, в UNIX, при помощи языка shell, можно создавать пакетные файлы, их еще называют скрипты (script). Исполнение командного файла мало отличается от исполнения команд в интерактивном режиме.

Если в строке встречается **"#'**, то с этого места до конца строки все символы являются комментарием.

" ;" означает перевод строки, служит для разделения команд. Можно писать несколько команд в одной строке, разделяя их **" ; "**.

Если в начале файла стоит **"#!"**, то за ним должно стоять имя оболочки **shell**, под которым будет работать этот командный файл.

#!/bin/sh - каким интерпретатором хотим исполнять этот файл. Возможная альтернатива: передать командный файл в качестве аргумента оболочки **shell**: **sh myfile**.

Для исполнения командного файла **myfile** можно сделать его выполняемым с помощью команды **chmod 755 myfile.** Его также можно выполнить, вызвав явно команду "sh" ("shell") ***sh myfile*** или ***sh < myfile.***

Файл можно выполнить и в текущем экземпляре "shell". Для этого существует специфическая команда "." (точка), то есть . ***myfile.***

1.1.3. Метасимволы

Иногда возникают задачи, для которых значение отдельных параметров неизвестно, или требуется выполнить схожие действия для большого числа операндов. Например, необходимо найти файл, первый символ имени которого неизвестен, или необходимо выполнить копирование всех файлов с расширением .txt. Для этих задач удобно использовать специальные символы (метасимволы) посредством которых упрощается задача указания файлов или групп файлов как аргументов команды.

Метасимвол «*» осуществляет поиск любой строки символов, включая нулевую (пустую) строку. Можно использовать «*» для обозначения полного или частичного имени файла. Просто символ «*» ищет все имена файлов и справочников в текущем справочнике, за исключением тех, которые начинаются с точки.

Метасимвол «?» осуществляет поиск любого одного символа в имени файла за исключением лидирующей точки. Предположим, вы имеете книгу, в которой 12 глав и хотите получить список глав до 9-ой главы. Если ваш справочник содержит следующие файлы:

Chapter1
Chapter2
Chapter5
Chapter9
Chapter11

то введите команду *ls* с метасимволом *?* для получения всех глав, которые начинаются со строки "*Chapter*" и заканчиваются одним символом:

$ ls Chapter?
Chapter1
Chapter2
Chapter5
Chapter9

Чтобы получить список всех глав в текущем справочнике, используйте метасимвол «*»:

*ls Chapter**

Если вы хотите найти любой символ из перечисленных вами символов, то заключите эти символы в квадратные скобки. Предположим, ваш справочник содержит следующие файлы: *cat, fat, mat, rat*. Если вы воспользуетесь в качестве части имени файла шаблоном *[crf]*, то будут найдены имена файлов, в которые входят либо буква "*c*", либо буква "*r*", либо буква "*f*" в указанной позиции. Пример:

$ ls [crf]at
cat
fat
rat
$

Символы, которые могут быть сгруппированы в скобки, называются классом символов.

Скобки могут также использоваться для обозначения диапазона символов, цифр или букв. Предположим в вашем справочнике содержатся следующие файлы: *chapter1, chapter2, chapter3, chapter4, chapter5, chapter6*. Если вы укажете:

chapter[1-5]

то будут найдены файлы с *chapter1* по *chapter5*.

Класс символов можно также указать с помощью диапазона букв. Если вы укажете *[A-Z],* то будут найдены только большие буквы, если *[a-z*] - то малые буквы.

Данные метасимволы можно использовать как в DOS так и в Unix

1.2. Практический блок

Задание 1. *Основы работы в DOS*

1. Вызовите командный процессор ***cmd.exe*** (*Пуск-Выполнть-cmd* или *Пуск-Все программы- Командная строка*).

2. Создайте на диске **D:** каталог своей группы

3. В своем каталоге создать каталоги ***First*** и ***Second***.

4. Работа в каталоге ***First***:

- Создать текстовый файл – содержащий описание способов создания файлов.

- Создать текстовый файл в текстовом редакторе MS DOS – определение операционной системы.

- Создать текстовый файл содержащий справочную информацию о текстовом редакторе (справка о команде).

5. В каталоге ***Second***:

- Скопировать все файлы из каталога ***First***

- Скопировать все файлы начинающиеся на ***A*** (латинскую) из каталога ***C:\WINDOWS***.

- Объединить файлы, скопированные из каталога ***First*** в один файл.

- Создать текстовый файл, отображающий содержимое каталога.

- Удалить все файлы, начинающиеся на ***A.***

6. Очистить экран.

7. Выведите на экран дерево каталогов своей папки.

Задание 2. *Работа с пакетными файлами DOS*

1. Создайте командный файл ***COMM.BAT***, выполняющий действия:

- Отключение режима отображения на экране выполняемой команды.

- Вывод на экран: "Копирование и удаление файла".

- Создание на диске каталога ***DIR1***, и в нем создание каталога ***DIR2***.

- Копирование файла с именем ***TEXT1.TXT***, предварительно созданного в папке группы, в файл с именем ***TEXTNEW.TXT*** в каталоге ***\DIR1\DIR2***.

- Удаление исходного файла.

- Вывод на экран: “Файл скопирован и удален”.

- Пауза до нажатия клавиши.

2. Создайте командный файл с именем ***SUMMA.BAT***, выполняющий действия:

- Вывод на экран «Объединение и переименование файлов».

- Объединение содержимого файлов ***A.TXT*** и ***B.TXT***, предварительно созданных в папке своей группы, под именем ***C.TXT***.

- Вывод содержимого объединенного файла на экран.

- Ожидание нажатия клавиши.

- Переименование файлов ***A.TXT*** и ***B.TXT*** в ***FINA.TXT*** и ***FINB.TXT*** соответственно.

- Вывод на экран: "Задание выполнено".

3. Предъявите работу преподавателю, удалите все созданные вами файлы и каталоги.

Задание 3. *Основы работы в shell*

1. Откройте окно терминала (в зависимости от дистрибутива доступ к терминалу может осуществляться разными способами, чаще всего *Главное меню-Стандартные-Терминал*).

2. Создайте в домашнем каталоге каталог, названный вашей фамилией.

3. В своем каталоге создать каталоги ***First*** и ***Second***.

4. Работа в каталоге ***First***:

- Создать текстовый файл, содержащий описание команды ***chmod***.

- Создать текстовый файл, содержащий описание команды ***test***.

5. В каталоге ***Second***:

- Скопировать все файлы из каталога ***First***

- Объединить файлы, скопированные из каталога ***First*** в один файл (используйте ***ca***t и перенаправление вывода в файл).

- Удалить файлы, скопированные из каталога ***First.***

6. Очистить экран.

Задание 4. *Работа с пакетными файлами shell*

1. Создайте командный файл ***COMM.sh***, выполняющий действия:

- Вывод на экран: "Копирование и удаление файла".

- Создание в домашнем каталоге каталога ***DIR1***, и в нем создание каталога ***DIR2.***

- Копирование файла с именем ***TEXT1.TXT***, предварительно созданного в папке группы, в файл с именем ***TEXTNEW.TXT*** в каталоге ***/DIR1/DIR2***.

- Удаление исходного файла.

- Вывод на экран: “Файл скопирован и удален”.

- Пауза до нажатия клавиши (для выполнения этого действия можно воспользоваться командой **read**).

2. Создайте командный файл с именем ***SUMMA.sh***, выполняющий действия:

- Вывод на экран “Объединение и переименование файлов”.

- Объединение содержимого файлов ***A.TXT*** и ***B.TXT***, предварительно созданных в папке своей группы, под именем ***C.TXT***.

- Вывод содержимого объединенного файла на экран.

- Ожидание нажатия клавиши.

- Переименование файлов ***A.TXT*** и ***B.TXT*** в ***FINA.TXT*** и ***FINB.TXT*** соответственно.

- Вывод на экран: "Задание выполнено".

Предъявите работу преподавателю, удалите все созданные вами файлы и каталоги.

2. Переменные, условия и циклы в пакетных файлах

2.1. Теоретический блок

2.1.1. DOS

Создавать серьезные программы при помощи пакетных файлов *DOS* не возможно, однако основные конструкции языков программирования в пакетных файлах присутствуют. Приведем их описание.

Переменные

Переменные в пакетных файлах *DOS* задаются при помощи команды **SET**. При использовании значения переменной, ее имя необходимо заключать в значки «**%**» с обеих сторон.

Пример. Хранение параметра в переменной

```
set file=dostoansi
C:\MASM32\BIN\Ml.exe /c /coff %file%.asm
C:\MASM32\BIN\Link.exe /SUBSYSTEM:WINDOWS %file%.obj /RELEASE
```

В переменных можно хранить не только строковые, но и числовые значения. Вычисление осуществляется командой **SET**, если она выполняется с ключом **/A**. При этом поддерживается практически полный набор операторов языка *C*, включая шестнадцатеричный модификатор *0x*.

Пример. Сложение двух чисел

```
@echo off
set a=2
set b=2
set /a c=%a%+%b%
echo %c%
pause
```

Чтобы организовать ввод значения переменной с клавиатуры, можно воспользоваться командой **SET** с параметром **/P**. Модифицируем предыдущий пример так, чтобы значения переменных *a* и *b* можно было вводить с клавиатуры.

Пример. Ввод значений переменных с клавиатуры

```
@echo off
set /p a="Input a "
set /p b="Input b "
set /a c=%a%+%b%
echo %c%
pause
```

Параметры командной строки

Иногда удобней вводить значения переменных не во время исполнения программы, а сразу передавать их на вход программы как параметр. Так например, при выполнении команды **DIR /P**, ключ **/P** является параметром и указывает что, необходимо выводить содержимое каталога постранично.

В командном файле параметры указываются с помощью выражений **%x**, где **x** - цифра от 1 до 9. **%0** означает имя самого командного файла. Разберем применение параметров командной строки на примере сложения двух чисел.

Результат работы программы приведен на рисунке 2.1.

Пример. Параметры командной строки

```
@echo off
set /a c=%1+%2
echo %c%
pause
```

Рисунок 2.1 - Работа программы

Ветвления в пакетных файлах DOS

Ветвления в пакетных файлах организуются при помощи команды **IF**. Оператор **IF** проверяет истинность какого-либо <*условия*> и в зависимости от этого выполняет или не выполняет команду <*следствие*>.

IF EXIST TEST.BAT GOTO lab1

если существует файл *TEST.BAT*, то перейти на метку *lab1*.

IF NOT EXIST TEST.BAT GOTO exit

если файл *TEST.BAT* не существует, то следует перейти на метку *exit*.

IF %1==ASDF GOTO asdflabel

если переменная *%1* равна *ASDF*, то осуществляется переход на метку *asdflabel*

IF ERRORLEVEL 1 GOTO err1

если ошибка (код возврата) выполнения предыдущей команды равно 1, то перейти на метку *err1*.

Если после проверки условия необходимо выполнить несколько команд, то их необходимо заключить в круглые скобки

```
if %1 == 0 (
set RESULT=1
exit /b
)
```

Также можно организовать ветвления с двумя и более направлениями

```
if condition (
ветвь 1
```

```
) else (
ветвь 2
)
```

Пример. Изменение цвета интерфейса в зависимости от параметра. Результаты работы программы приведены на рисунке 2.2.

```
@echo off
if %1==r (
color 04
goto ext
) else (
color 40
goto ext
)
:ext
Pause
```

а

б

Рисунок 2.2 – Результаты работы программы: а – ветвь 1, б – ветвь 2

Циклы в пакетных файлах DOS

Команда **FOR** позволяет организовать выполнение повторяющихся однотипных действий. Можно использовать ее для того, чтобы вывести на экран числа от одного до десяти.

```
for /l %%i in (1,1,10) do echo %%i
```

Переменная *i* называется счетчиком цикла. В силу своеобразия синтаксиса команды **FOR**, имя счетчика цикла должно состоять из одной буквы. Причем, если мы пишем командный файл, то перед именем счетчика цикла надо поставить сдвоенный знак процента, если же мы просто набираем команду в командной строке, то одиночный.

Логика работы этой команды такова. После слова *in* указан диапазон изменения счетчика цикла. В данном варианте команды это тройка чисел: начальное значение счетчика, шаг счета, предельное значение счетчика. При выполнении команды командный процессор сначала присвоит переменной *i* значение *1*, а потом на каждом шаге цикла будет увеличивать его на *1*, пока оно не превысит *10*. Очевидно, таких шагов получится десять. Если бы в качестве шага счета мы указали число *2*, то цикл выполнился бы пять раз. На каждом шаге цикла выполняется тело цикла, написанное после слова *do*. В

приведенном примере это команда *echo*, которая выводит на экран текущее значение счетчика цикла.

Обычно команда **FOR** используется для перебора и обработки файлов. Надо сказать, что в достаточно простых случаях массовая обработка файлов выполняется с помощью подстановочных символов. Если, мы хотим всем файлам в текущем каталоге заменить расширение *.htm* на *.html*, мы вводим команду *ren *.htm *.html*. Но если то же самое надо сделать не в одном каталоге, а в дереве каталогов, то без команды **FOR** не обойтись. Приведенный ниже командный файл выполняет эту операцию для всех htm-файлов в подкаталоге *website* текущего каталога. Точнее, во всем дереве каталогов, которое находится внутри *website.*

```
for /r website %%i in (*.htm) do ren %%i %%~ni.html
```

Ключ **/r** указывает на необходимость обхода каталога *website* и всех его внутренностей. Если его не указать (но тогда и каталог указывать не разрешается), то обработаны будут только файлы в текущем каталоге. Диапазоном значений счетчика цикла в данном варианте команды является множество всех файлов с расширением *.htm,* находящихся внутри каталога (точнее, дерева) *website*. Запись *~ni* означает, что из значения переменной *i* требуется выделить только имя файла. В языке команд MS-DOS предусмотрено несколько таких модификаторов, например, запись *~xi* обозначает расширение файла. Все модификаторы описаны в справке по команде **FOR**.

Тело цикла может состоять из нескольких команд, заключенных в скобки.

```
@echo off
for /r website %%i in (*.htm) do (
  rem Выводим имя файла
  echo %%i
  rem Переименовываем файл
  ren %%i %%~ni.html
)
```

2.1.2. UNIX

Переменные

Все переменные в языке *shell* - текстовые. Их имена должны начинаться с буквы и состоять из латинских букв, цифр и знака подчеркивания. Чтобы воспользоваться значением переменной, надо перед ней поставить символ «**$**». Использование значения переменной называется подстановкой.

Начальные значения переменным с именем могут быть установлены следующим образом:

<имя>=<значение> [<имя>=<значение>] ...

Не может быть одновременно функции и переменной с одинаковыми именами. Для подстановки значений переменных возможны также следующие конструкции:

${<переменная>}

если значение <*переменной*> определено, то оно подставляется. Скобки применяются лишь если за <*переменной*> следует символ, который без скобок приклеится к имени.

Для проведения арифметических вычислений можно воспользоваться командой **let.** В большинстве случаев, **let** можно считать упрощенным вариантом команды **expr. expr** - универсальный обработчик выражений: вычисляет заданное выражение (аргументы должны отделяться пробелами). Выражения могут быть арифметическими, логическими или строковыми.

Пример. Сложение двух чисел

```
a=1
b=2
let c=$a+$b ‘ c=`expr $a+$b`
echo $c
echo нажмите любую клавишу
read
```

Для организации ввода переменных с клавиатуры используется команда **read**

read [<переменная> ...]

Читается из стандартного ввода одна строка; первое ее слово присваивается первой переменной, второе - второй и т.д., причем все оставшиеся слова присваиваются последней переменной.

Пример. Ввод значений переменных из командной строки

```
echo введите a
read a
echo введите b
read b
c=`expr $a+$b`
echo $c
echo нажмите любую клавишу
read
```

Параметры командной строки

Аргументы командной строки в shell-скриптах задаются точно также как в пакетных файлах DOS, только при обращении к ним используется знак «**$**».

Пример. Параметры командной строки. Результат работы программы приведен на рисунке 2.3.

```
let c=$1+$2
echo $c
```

echo нажмите любую клавишу
read

```
victim@linux-edia:~> ./1.sh 1 2
3
нажмите любую клавишу
```

Рисунок 2.3 – Параметры командной строки в Linux

Ветвления в shell

1. Оператор ***if***

if <список1>
then
<список2>
[elif <список3>
then
<список4>]
. . .
[else
<список5>]
fi

Выполняется *<список1>* и, если код его завершения *0*, то выполняется *<список2>*, иначе - *<список3>* и, если и его код завершения *0*, то выполняется *<список4>*. Если же это не так, то выполняется *<список5>*. Части *elif* и *else* могут отсутствовать.

2. Оператор ***case***

case $<переменная> in
<шаблон> | <шаблон>...) <список> ;;
. . .
esac

Оператор выбора выполняет *<список>*, соответствующий первому *<шаблону>*, которому удовлетворяет *<переменная>*. Форма шаблона та же, что и используемая для генерации имен файлов. Часть | шаблон... может отсутствовать.

Циклы в shell

1. Цикл ***for***

for <переменная> [in <набор>]
do
<список>
done

Если часть *in <набор>* опущена, то это означает *in "$@"* (то есть *in $1 $2 ... $n)*.

Пример. Вывести на экран содержимое всех текстовых файлов текущего каталога:

```
for f in *.txt
do
cat $f
done
```

2. Цикл ***while (until)***

```
while <список1>
do
<список2>
done
```

До тех пор, пока код завершения последней команды <*списка1*> есть *0*, выполняются команды <*списка2*>. При замене служебного слова **while** на **until** условие выхода из цикла меняется на противоположное.

В качестве одной из команд <*списка1*> может быть команда **true (false)**. По этой команде не выполняется никаких действий, а код завершения устанавливается *0 (-1)*. Эти команды применяются для организации бесконечных циклов. Выход из такого цикла можно осуществить лишь по команде **break.**

2.2. Практический блок

Задание 1. *Работа с переменными*

Напишите пакетный файл для DOS, сравнивающий два числа. Числа вводятся пользователем с клавиатуры. В результате сравнения выводится одно из следующих сообщений: «Первое число больше», «Второе число больше», «Числа равны».

Задание 2. *Параметры командной строки*

Напишите пакетный файл для DOS, определяющий наличие файла в директории с вводимым пользователем именем. Имя искомого файла передается как параметр командной строки. В случае если файл присутствует в директории, то ему устанавливается атрибут *«системный»*; иначе выводится сообщение *«файл не найден»*. Для установления атрибутов файла воспользуйтесь командой **ATTRIB**, синтаксис команды разберите самостоятельно.

Задание 3. *Организация цикла*

Создайте пакетный файл, выводящий имя и содержимое всех файлов с расширением *.txt* в задаваемой пользователем директории. Каталог задается из командной строки. Перед выводом содержимого каждого файла выводится полное имя файла (включая каталог).

Задание 4. *Организация меню*

Создайте пакетный файл, выполняющий резервное копирование каталога. Имя резервного каталога задается пользователем, если пользователь не задает имя каталога, то копия помещается в том же разделе, где исходный каталог. Пакетный файл должен предоставлять пользователю меню, позволяющее выбрать один из следующих режимов работы:

1) замещение файлов (файлы из исходного каталога замещают файлы резервного);

2) добавление файлов (если в резервном каталоге файла не существует, то он добавляется из исходного, прочие файлы не заменяются);

3) удаление файлов (если в резервном каталоге существует файл, которого нет в исходном, то он удаляется).

Предусмотрите возможность вызова справки по работе пакетного файла.

Задание 5. *Работа с переменными shell*

Напишите скрипт для shell, сравнивающий два числа. Числа вводятся пользователем с клавиатуры. В результате сравнения выводится одно из следующих сообщений: *«Первое число больше», «Второе число больше», «Числа равны»*. Для сравнения чисел используйте команду **test.**

Задание 6. *Параметры командной строки в скриптах shell*

Напишите скрипт для shell, определяющий наличие файла в директории с вводимым пользователем именем. Имя искомого файла передается как параметр командной строки. В случае если файл присутствует в директории, случае если файл присутствует в директории, выводится следующая информация о файле: доступен ли файл на чтение, запись, исполнение; иначе выводится сообщение *«файл не найден»*. Для получения информации об атрибутах файла воспользуйтесь командой **test**, синтаксис команды разберите самостоятельно.

Задание 7. *Организация цикла в скриптах shell*

Создайте пакетный файл, выводящий имя и содержимое всех файлов с расширением *.txt* в задаваемой пользователем директории. Каталог задается из командной строки. Перед выводом содержимого каждого файла выводится полное имя файла (включая каталог).

Задание 8. *Резервное копирование файлов*

Создайте скрипт, выполняющий резервное копирование каталога. Имя резервного каталога задается пользователем, если пользователь не задает имя каталога, то копия помещается в том же разделе, где исходный каталог. В зависимости от введенного пользователем параметра командной строки, скрипт должен предусматривать следующие режимы работы:

1) замещение файлов (файлы из исходного каталога замещают файлы резервного);

2) добавление файлов (если в резервном каталоге файла не существует, то он добавляется из исходного, прочие файлы не заменяются);

3) удаление файлов (если в резервном каталоге существует файл, которого нет в исходном, то он удаляется);

4) синхронизация файлов (сравниваются даты последней модификации двух файлов, более новый файл копируется в резервный каталог).

Все режимы работы должны предусматривать рекурсивный обход вложенных каталогов. Предусмотрите возможность вызова справки по работе пакетного файла. Для сравнения времен модификации файлов используйте команду **test**.

3. Управление системой Linux при помощи командных файлов

3.1. Теоретический блок

3.1.1. Файловая система

Файловой системой называется некоторая организация данных и метаданных на устройстве хранения.

Все файлы в Unix физически состоят из двух частей, реально локализованных в различных блоках дискового накопителя, но обязательно находящихся в одном дисковом разделе, первичном или логическом.

Первая часть файла - его так называемые метаданные, которые содержат файловый дескриптор (это просто некое уникальное число), сведения о его атрибутах (принадлежности, правах доступа, времени модификации и т.д.), а также информацию о том, в каких блоках дискового раздела (которые так и называются - блоки данных) физически размещено содержимое файла - его вторая часть.

Все файлы, доступные в Unix системах, составляют иерархическую файловую структуру, которая подобна растущему дереву имеет ветки (каталоги) и листья (файлы в каталогах). Корень этого большого дерева обозначается как */*. Физически файлы могут располагаться на различных устройствах.

Команды mount и umount

Команда **mount** служит для подключения файловых систем разных устройств к этому большому дереву. Также существует противоположная ей команда под названием **umount**, которая выполняет демонтирование (отключение) файловых систем.

Наиболее часто встречающаяся форма команды **mount** выглядит следующим образом:

mount -t vfstype device dir

Такая команда предлагает ядру смонтировать (подключить) файловую систему указанного типа *vfstype*, расположенную на устройстве *device*, к заданному каталогу *dir*, который часто называют точкой монтирования. Предыдущее содержимое, владелец и режим доступа к каталогу *dir* становятся недоступными (исчезают), а вновь появившиеся продолжают действовать, пока файловая система *device* смонтирована (подключена) к *dir*.

Программы **mount** и **umount** поддерживают список текущих смонтированных файловых систем в файле */etc/mtab*. Запущенная без аргументов, **mount** выводит на экран этот список.

Подробности работы команд **mount** и **umount**, а также список их параметров приведены в справочной системе Linux.

Команда df

df —показывает список всех файловых систем по именам устройств, сообщает их размер, занятое и свободное пространство и точки монтирования.

Подробно работа команды **df** описана в справочной системе Linux.

Команда du

du выдает отчет об использовании дискового пространства заданными файлами, а также каждым каталогом иерархии подкаталогов каждого указанного каталога. Здесь под использованным дисковым пространством понимается пространство, используемое для всей иерархии подкаталогов указанного каталога.

Запущенная без аргументов, команда **du** выдает отчет о дисковом пространстве для текущего каталога.

Подробности о работе команды **du** смотрите в справочной системе Linux.

i-node

При создании файловой системы создаются также и структуры данных, содержащие информацию о файлах. Каждый файл имеет свой инод, идентифицируемый по номеру инода (часто называемый 'i-номером' или 'инодом'), в файловой системе, в которой располагается сам файл.

Иноды хранят информацию о файлах, такую как принадлежность владельцу (пользователю и группе), режим доступа (чтение, запись, запуск на выполнение) и тип файла. Существует определенное число инодов, которое указывает максимальное количество файлов, допускаемое определенной файловой системой. Обычно, при создании файловой системы примерно 1% ее выделяется под иноды.

Номер инода файла можно посмотреть используя команду **ls -i**, а команда **ls -l** покажет информацию, хранящуюся в иноде.

Жесткие и символьные ссылки

Жёсткой ссылкой в UNIX-подобных операционных системах называется имя файла, привязанное к уникальному индексному дескриптору файла. Таким образом, понятия «жёсткая ссылка на файл» и «имя файла» являются синонимами. Создать жёсткую ссылку в UNIX-подобных ОС можно при помощи команды **ln**, которая по умолчанию создаёт именно жёсткие ссылки.

В метаинформации файла всегда хранится число жёстких ссылок на него (то есть количество его имён). Файл считается удалённым после удаления его последнего имени, однако место освобождается лишь когда его индексный дескриптор перестаёт использоваться.

Символьная ссылка (*Symbolic link*) — специальный файл в UNIX-подобных операционных системах, для которого в файловой системе не хранится никакой информации, кроме одной текстовой строки. Эта строка трактуется как путь к файлу, который должен быть открыт при попытке обратиться к данной ссылке. Символьная ссылка занимает ровно столько места на файловой системе, сколько требуется для записи её содержимого (нормальный файл занимает как минимум один блок раздела).

Целью ссылки может быть любой объект — например, другая ссылка, или даже несуществующий файл (в последнем случае при попытке открыть его должно выдаваться сообщение об отсутствии файла). Ссылка, указывающая на несуществующий файл, называется *висячей*.

Практически символьные ссылки используются для более удобной организации структуры файлов на компьютере, так как позволяют одному файлу или каталогу иметь несколько имён и свободны от некоторых ограничений, присущих жёстким ссылкам (последние действуют только в пределах одного раздела и не могут ссылаться на каталоги).

Команда ln -s файл1 файл2 создает символьную ссылку *файл2* на *файл1*. При этом *файл1* может и не существовать. Символическая ссылка с именем *файл2* всё равно будет создана.

Команда chmod

chmod — изменение прав доступа к файлам и папкам. Права записываются сразу для трёх типов пользователей: владельца файла, группы, в которую он входит и для прочих пользователей. **chmod** может быть записан в двух форматах: в числовом и в символьном.

Пример символьной записи: *'rwxr-xr-x'*. **chmod +w {имяфайла}** добавит права записи для владельца файла**, chmod -w {имяфайла} –** выполняет обратное действие.

Примером числовой записи может служить *'755*', которая эквивалентна записанной выше строковой записи: каждое право имеет числовой код и может быть задано вручную:

- 400 — владелец имеет право на чтение;
- 200 — владелец имеет право на запись;
- 100 — владелец имеет право на выполнение;
- 40 — группа имеет право на чтение;
- 20 — группа имеет право на запись;
- 10 — группа имеет право на выполнение;
- 4 — остальные имеют право на чтение;
- 2 — остальные имеют право на запись;
- 1 — остальные имеют право на выполнение.

Суммировав эти коды можно получить символьную запись. Например, **chmod 444 {имяфайла}**: 400+40+4=444 — все имеют право только на чтение.

umask

umask — функция, изменяющая права доступа, которые присваиваются новым файлам и директориям по умолчанию. Права доступа файлов, созданных при конкретном значении **umask**, вычисляются при помощи следующих побитовых операций (**umask** обычно устанавливается в восьмеричной системе счисления): *побитовое И* между унарным дополнением аргумента (используя *побитовое НЕ*) и режимом полного доступа.

Режим полного доступа для файлов — *666*, для директорий — *777*.

Фактически, **umask** указывает, какие биты следует сбросить в выставляемых правах на файл - каждый установленный бит **umask** запрещает выставление соответствующего бита прав. Исключением из этого запрета является бит исполняемости, который для каталогов выставляется особо. **umask 0** означает, что следует (можно) выставить все биты прав (*rwxrwxrwx*), **umask 777** запрещает выставление любых прав.

Примеры

Допустим, что значение **umask** равняется *174*, тогда каждый новый файл будет иметь права доступа *602*, а каждая новая директория *603*:

666_8 *И* $НЕ(174_8) = 602_8$

и

777_8 *И* $НЕ(174_8) = 603_8$

$777_8 = (111\ 111\ 111)_2$

$174_8 = (001\ 111\ 100)_2$

$НЕ(001\ 111\ 100)_2 = (110\ 000\ 011)_2$

$(111\ 111\ 111)_2$ *И* $(110\ 000\ 011)_2 = (110\ 000\ 011)_2$

777_8 $НЕ\ (174)_8$ $(603)_8$

В bash:

```
$ umask 0174
$ mkdir директория
$ touch файл
$ ls -l
drw-----wx 2 dave dave 512 Sep  1 20:59 директория
-rw-----w- 1 dave dave   0 Sep  1 20:59 файл
```

3.1.2. Пользователи

Файл /etc/passwd

/etc/passwd — файл, содержащий в текстовом формате список пользовательских учётных записей. Является первым и основным

источником информации о правах пользователя операционной системы. Существует в большинстве версий и вариантов UNIX-систем.

Каждая строка файла описывает одного пользователя и содержит семь полей, разделённых двоеточиями:

1) регистрационное имя, или логин;
2) хеш пароля;
3) идентификатор пользователя;
4) идентификатор группы по умолчанию;
5) информационное поле GECOS;
6) начальный (он же домашний) каталог;
7) регистрационная оболочка, или shell.

Основным назначением **/etc/passwd** является сопоставление логина и идентификатора пользователя (*UID*). Изначально поле пароля содержало хеш пароля и использовалось для аутентификации. Однако в связи с ростом вычислительных мощностей процессоров появилась серьёзная угроза применения простого перебора для взлома пароля. Поэтому все пароли были перенесены в специальные файлы, такие как */etc/shadow* в GNU/Linux или */etc/master.passwd* во FreeBSD. Эти файлы недоступны для чтения обычным пользователям. Поле *GECOS* хранит вспомогательную информацию о пользователе (номер телефона, адрес, полное имя и так далее). Оно не имеет чётко определённого синтаксиса.

Команда finger

finger [-bfilpqsw] [login1 [login2 ...]]

По умолчанию команда **finger** выводит в список для каждого пользователя системы UNIX на данный момент имя регистрации в систему, полное имя, имя терминала и статус записи (при отсутствии разрешения на запись перед терминальным именем указывается символ "*"), время простоя, время регистрации, нахождение места работы и телефонный номер (если они известны). (Время простоя отображается в минутах, если оно выведено одним целым числом, в часах и минутах, если в его отображении присутствует двоеточие (:), или в днях и часах, если в выводе присутствует символ *"d"*.)

Кроме того, существует более длинный формат вывода и он используется командой **finger** в том случае, если задан список имен пользователей. (Допускаются наряду с первым и последним именами пользователей также и учетные имена.) Этот формат состоит из нескольких строк; он включает всю информацию, описанную выше, и, дополнительно, пользовательские входной каталог и интерпретатор shell регистрации, любой план, который пользователь разместил в файле *.plan* в своем входном каталоге, и проект, в соответствии с которым заданные пользователи работают в данный момент, взятый из файла *.project*, который также находится во входном каталоге. Если в домашней директории указанного

пользователя находится файл *.nofinger*, то по команде **finger** информация об этом пользователе не возвращается.

Подробности о работе команды **finger** смотрите в справочной системе Linux.

Команда who

who [-u] [-T] [-l] [-H] [-q] [-p] [-d] [-b] [-r] [-t] [-a] [-s] [файл]

Команда **who** сообщает имя пользователя, имя терминальной линии, астрономическое время начала сеанса, продолжительность бездействия терминальной линии с момента последнего обмена, идентификатор процесса интерпретатора команд shell для каждого из пользователей, работающих в системе UNIX. Для получения этой информации команда просматривает файл */etc/utmp*. Впрочем, вместо него может просматриваться другой файл, имя которого должно быть тогда указано в командной строке (файл должен иметь формат *utmp*). Обычно в качестве файла указывают */etc/wtmp*, где зафиксированы времена начала всех сеансов с момента его последнего создания.

Команда **whoami** идентифицирует обратившегося с ней пользователя.

Выдаваемые сообщения имеют, вообще говоря, следующий формат:

NAME [STATE] LINE TIME [IDLE] [PID] [COMMENT] [EXIT]

Информация *NAME, LINE* и *TIME* выдается при всех опциях, кроме **-q**; *STATE* - только при **-T;** *IDLE* и *PID* - только при **-u** и **-l**; и, наконец, *COMMENT* и *EXIT* - только при **-a**. Какая информация выдается в случае опций **-p, -d** и **-r**, подробно обясняется для каждой из них отдельно.

Задавая различные опции, с помощью команды **who** можно получить информацию о времени начала и конца сеансов, перезагрузок, корректировках системных часов, а также о других процессах, порожденных процессом *init.*

Подробности о работе команды **who** смотрите в справочной системе Linux.

Команда id

id - утилита, выводящая информацию об указанном пользователе или текущем пользователе, который запустил данную команду и не указал явно имя пользователя. По умолчанию выводятся подлинные числовые идентификаторы пользователя (*UID*) и группы (*GID*), действующие (именные) идентификаторы пользователей и групп, а также идентификаторы других групп, в которых состоит пользователь.

Подробно работа команды **id** описана в справочной системе Linux.

3.1.3. Процессы

Каждая выполняемая программа называется процессом. Эти процессы варьируются от вещей типа X Window System до системных программ (демонов), которые запускаются во время загрузки системы. Каждый процесс выполняется от имени какого-либо пользователя. Процессы, запускаемые во время загрузки, обычно выполняются от имени *root* или *nobody*. Процессы, запускаемые пользователями, работают с правами этих пользователей.

Пользователь имеет полный контроль, над процессами, которые он запустил. Кроме того *root* может управлять всеми процессами в системе, включая те, что были запущены другими пользователями. Процессами можно управлять и наблюдать за ними с помощью различных программ, а также с помощью некоторых команд shell.

Приоритетные и фоновые задачи

Программы, запускаемые из командной строки, начинают работать в приоритетном режиме (*foreground*). Это позволяет пользователю видеть данные, выводимые программой, и взаимодействовать с ней. Однако в некоторых случаях хотелось бы, чтобы программа работала, не захватывая полностью терминал. Это называется работой программы в фоновом режиме (*background*), и для осуществления этой задачи существует несколько способов.

Первым способом перевода процесса в фоновый режим является добавление в командную строку амперсанда (**&**) при запуске программы. Например, допустим, воспользуемся консольным mp3-плеером *amp* для воспроизведения mp3-файлов, однако в то же время нужно работать в том же самом терминале. Следующая команда запустит *amp* в фоновом режиме:

*% amp *.mp3 &*

Программа запустится и будет работать как обычно, а пользователь снова вернётся в командную строку.

Другим способом перевода процесса в фоновый режим является осуществление этого во время работы процесса. Сначала запустим программу. Во время её работы нажмем **Control+z**. При этом процесс будет приостановлен. Иными словами приостановленный процесс “ставится на паузу”. Он мгновенно прекращает свою работу, но может быть вновь запущен в любое время. После того, как пользователь приостановили процесс, он возвращаетесь в командную строку. Затем можно перевести процесс в фоновый режим, набрав **bg.** При этом приостановленный процесс вновь продолжит свою работу, но уже в фоновом режиме.

Если необходимо взаимодействовать с фоновым процессом, то можно снова перевести его в приоритетный режим. Если выполняется только один фоновый процесс, можно переключиться на него, набрав **fg.** Если программа не завершила своё выполнение, она возьмёт под свой контроль терминал и

будет невозможно получить доступ к приглашению командной строки. Иногда программа завершает своё выполнение, работая в фоновом режиме. В этом случае будет получено сообщение типа:

[1]+ Done /bin/ls $LS_OPTIONS

Это означает, что фоновый процесс (в данном случае *ls*) завершил свою работу.

Одновременно могут выполняться несколько фоновых процессов. В этом необходимо выяснить, какой именно процесс нужно перевести назад в приоритетный режим. Просто набрав **fg**, будет возвращен назад процесс, который был последним переведён в фоновый режим. Для вывода перечня всех заданий существует команда **jobs**.

```
% jobs
[1]   Stopped              vim
[2]-  Stopped              amp
[3]+  Stopped              man ps
```

По сути, команда выводит список всех процессов, переведённых в фоновый режим. В данный момент, все они остановлены (*stopped*). Это означает, что их работа этих процессов приостановлена. Числа слева - это идентификаторы, согласно которым сортируются все фоновые процессы. Идентификатором с плюсом отмечен процесс, который будет переведён в приоритетный режим, если будет набрано просто **fg**.

Если нужно переключиться в *vim*, то необходимо набрать:

% fg 1

и *vim* возьмёт консоль под свой контроль.

Перевод процессов в фоновый режиме весьма полезно, если у пользователя есть только один терминал через коммутируемое подключение.

Команда ps

Для того, чтобы получить перечень всех программ, выполняющихся в системе нужно воспользоваться командой **ps**. У этой команды есть много опций, поэтому мы рассмотрим только самые важные из них.

Чтобы получить список программ, выполняемых в терминале нужно просто набрать **ps** без параметров. В этот список входят процессы, работающие в приоритетном режиме (любой командный процессор, в котором работает пользователь, и сама команда **ps**). Также перечисляются фоновые процессы, которые могут выполняться в данный момент. В большинстве случаев это будет очень короткий список:

```
% ps
  PID TTY          TIME CMD
 7923 ttyp0    00:00:00 bash
 8059 ttyp0    00:00:00 ps
```

PID - это идентификатор процесса (*process ID*). Все работающие процессы имеют уникальные идентификаторы в диапазоне от *1* до *32767*. Каждому новому процессу присваивается следующий свободный *PID*. Когда процесс завершает свою работу (или убивается, как вы увидите в следующем разделе), он отдаёт свой *PID*. При достижении в системе максимального *PID*, следующим берётся первый свободный *PID* с наименьшим номером, и так по кругу.

Колонка *TTY* обозначает терминал, в котором выполняется процесс. При простом вызове **ps** будет выведен список только тех программ, которые выполняются в текущем терминале. Поэтому все процессы будут иметь одну и ту же информацию в колонке *TTY*. Как видно в примере, оба процесса в списке выполняются в терминале *ttyp0*. Это означает, что они выполняются или удалённо, или в каком-нибудь X-терминале.

Колонка *TIME* содержит данные о времени, в течение которого процесс использует ресурсы центрального процессора. Однако это не то время, в течение которого выполняется процесс.

В колонке *CMD* представлена сама программа. В ней отображается только имя программы, безо всяких опций командной строки или подобной информации. Для получения боле полной информации нужно использовать одну из множества опций команды **ps**.

Можно получить полный список процессов, выполняемых в вашей системе, используя правильный набор опций. Это приведёт к появлению большого списка процессов.

```
%ps –ax
PID TTY      STAT   TIME COMMAND
  1 ?        S      0:03 init [3]
  2 ?        SW     0:13 [kflushd]
  3 ?        SW     0:14 [kupdate]
  4 ?        SW     0:00 [kpiod]
  5 ?        SW     0:17 [kswapd]
 11 ?        S      0:00 /sbin/kerneld
 30 ?        SW     0:01 [cardmgr]
106 tty1     S      0:08 -bash
108 tty3     SW     0:00 [agetty]
109 tty4     SW     0:00 [agetty]
 111 tty6    SW     0:00 [agetty]
[вывод сокращён]
```

Большинство этих процессов запускаются во время загрузки на большинстве систем. Уязвимость в ядре в функции *ptrace* стала причиной исправления, в результате которого для многих выполняемых процессов больше не показываются опции командной строки. В данном примере названия этих процессов заключены в квадратные скобки, например у

процессов с *PID* от *108* до *111*. В этом листинге также появилось несколько новых колонок.

Во-первых, большинство процессов в списке работают в *tty "?"*. Они не привязаны ни к одному из терминалов. Это является самым распространённым случаем для демонов, т.е. процессов, которые выполняются без привязки к какому-либо терминалу.

Во-вторых, здесь появился новый столбец: *STAT*. В нём отображается состояние процесса. *S* означает, что процесс спит (*sleeping*), т.е. ожидает какого-либо события. *Z* означает процесс-зомби. Такой процесс появляется в том случае, когда умирает его родительский процесс, оставив после себя дочерний процесс. Это нехорошая ситуация. *D* означает процесс, который перешёл в "непробудный" сон. Часто такие процессы невозможно убить даже путём отправки им сигнала *SIGKILL*. *W* означает *paging* (страничную подкачку файлов). Мёртвые процессы обозначаются как *X*. Процессы, отмеченные как *T*, трассируются или остановлены. *R* означает, что процесс можно запустить.

Ещё более подробную информацию о выполняемых процессах можно получить, выполнив следующую команду:

% ps –aux

USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND

root 1 0.0 0.0 344 80 ? S Mar02 0:03 init [3]
root 2 0.0 0.0 0 0 ? SW Mar02 0:13 [kflushd]
root 3 0.0 0.0 0 0 ? SW Mar02 0:14 [kupdate]
root 4 0.0 0.0 0 0 ? SW Mar02 0:00 [kpiod]
root 5 0.0 0.0 0 0 ? SW Mar02 0:17 [kswapd]
root 11 0.0 0.0 1044 44 ? S Mar02 0:00 /sbin/kerneld
root 30 0.0 0.0 1160 0 ? SW Mar02 0:01 [cardmgr]
bin 50 0.0 0.0 1076 120 ? S Mar02 0:00 /sbin/rpc.port
[вывод сокращён]

Это практически вся информация о системе. В ней присутствует дополнительная информация о процессе: какой пользователь его запустил, сколько он использует системных ресурсов (колонки *%CPU, %MEM, VSZ* и *RSS*) и когда был запущен. Следует отметить ещё один момент: данные теперь выходят за пределы экрана, и вы не можете увидеть их полностью. Опция **-w** заставит **ps** переносить длинные строки.

Команда kill

Для принудительного завершения программы можно использовать команду **kill.** Также её можно использовать для управления процессами разными способами.

Чтобы убить процесс, нужно знать его *PID* или имя. Чтобы узнать идентификатор, можно воспользоваться командой **ps**, описанной в

предыдущем разделе. Например, чтобы убить процесс *4747*, нужно выполнить следующее:

% kill 4747

Чтобы убить процесс, нужно быть его владельцем. Это особенность системы безопасности. *root* может убить любой процесс в системе.

Существует ещё одна разновидность утилиты **kill** под название **killall**. Эта программа убивает все работающие процессы с заданным именем. Если нужно убить все процессы *vim*, можно набрать следующую команду:

% killall vim

Эту команду с правами администратора убьет все процессы *vim*, запущенные пользователями системы.

Иногда обычный **kill** не справляется с поставленной задачей. Определённые процессы не будут завершаться. Если *PID 4747* не отвечает на запрос **kill**, можно выполнить следующее:

% kill -9 4747

То же самое вы можете использовать и с **killall**. В данном случае процессу просто отправляется другой сигнал. Обычный вызов **kill** отправляет процессу сигнал *SIGTERM (terminate)*, который сообщает ему, что нужно остановить свою работу, сбросить буферы и выгрузить себя из памяти. **kill -9** отправляет процессу сигнал *SIGKILL (kill),* который по сути просто убивает его. Процессу не разрешается “чисто” завершить свою работу, и иногда это приводит к нежелательным последствиям, таким как повреждение данных.

Одним из полезных вариантов использования **kill** является перезапуск процесса. Отправка большинству процессов сигнала *SIGHUP (-1)* заставит их повторно прочитать свои конфигурационные файлы. Это особенно полезно для сообщения системным процессам о том, что им нужно перечитать свои конфигурационные файлы после их редактирования.

Команда top

Команда **top** позволяет просматривать постоянно обновляющуюся информацию о процессах, выполняющихся в системе. На экран выводится информация о выполняющихся в системе процессах плюс некоторая дополнительная информация о системе: средняя загрузка, количество процессов, состояние процессора, информация о свободной памяти. Также предоставляется подробная информация о процессах, включая *PID*, пользователя, приоритет, использование процессора и памяти, время работы и название программы.

6:47pm up 1 day, 18:01, 1 user, load average: 0.02, 0.07, 0.02

61 processes: 59 sleeping, 2 running, 0 zombie, 0 stopped

CPU states: 2.8% user, 3.1% system, 0.0% nice, 93.9% idle

Mem: 257992K av, 249672K used, 8320K free, 51628K shrd, 78248K buff

Swap: 32764K av, 136K used, 32628K free, 82600K cached

PID USER PRI NI SIZE RSS SHARE STAT LIB %CPU %MEM TIME COMMAND
112 root 12 0 19376 18M 2468 R 0 3.7 7.5 55:53 X
4947 david 15 0 2136 2136 1748 S 0 2.3 0.8 0:00 screenshot
3398 david 7 0 20544 20M 3000 S 0 1.5 7.9 0:14 gimp
4946 root 12 0 1040 1040 836 R 0 1.5 0.4 0:00 top
121 david 4 0 796 796 644 S 0 1.1 0.3 25:37 wmSMPmon
115 david 3 0 2180 2180 1452 S 0 0.3 0.8 1:35 wmaker
4948 david 16 0 776 776 648 S 0 0.3 0.3 0:00 xwd
1 root 1 0 176 176 148 S 0 0.1 0.0 0:13 init
189 david 1 0 6256 6156 4352 S 0 0.1 2.4 3:16 licq
4734 david 0 0 1164 1164 916 S 0 0.1 0.4 0:00 rxvt
2 root 0 0 0 0 0 SW 0 0.0 0.0 0:08 kflushd
3 root 0 0 0 0 0 SW 0 0.0 0.0 0:06 kupdate
4 root 0 0 0 0 0 SW 0 0.0 0.0 0:00 kpiod

Она называется **top** потому, что программы, наиболее активно использующие процессор, будут находиться вверху. Интересное замечание: сам **top** будет первым в списке в большинстве систем с низкой активностью вследствие своего использования процессора.

Если нужно выводить список только своих процессов или процессов какого-то другого пользователя, то нужные процессы могут отсутствовать среди тех, что активно используют процессор. Опция **-u** позволит вам указать имя пользователя или его *UID* и наблюдать только процессами, владельцем которых является этот *UID*.

% top -u alan
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
3622 alan 13 0 11012 10m 6956 S 1.0 2.1 0:03.66 gnome-terminal
3739 alan 13 0 1012 1012 804 R 0.3 0.2 0:00.06 top
3544 alan 9 0 640 640 568 S 0.0 0.1 0:00.00 xinit
3548 alan 9 0 8324 8320 6044 S 0.0 1.6 0:00.30 gnome-session
3551 alan 9 0 7084 7084 1968 S 0.0 1.4 0:00.50 gconfd-2
3553 alan 9 0 2232 2232 380 S 0.0 0.4 0:00.05 esd
3555 alan 9 0 2552 2552 1948 S 0.0 0.5 0:00.10 bonobo-activati

3.2. Практический блок

Задание 1. Создайте скрипт, который предоставляет пользователю следующие функции:

1. Создает отчет о количестве доступных файловых систем, их свободном и занятом пространстве;
2. Выводит информацию об имени и размере файла отчета, а также о правах пользователя на него.

Задание 2. Создайте скрипт, который предоставляет пользователю следующие функции:

1. Создает отчет об именах файлов их инодах, правах, размерах и временах последнего доступа для каталога, указанного пользователем (отчет формируется в порядке возрастания инодов);
2. Открывает полные права на файл отчета для всех пользователей;
3. Создает жесткую ссылку на файл отчета в домашнем каталоге пользователя и символьную ссылку на рабочем столе.

Задание 3. Создайте скрипт, который предоставляет пользователю следующие возможности:

1. Выясняет, зарегистрирован ли в системе пользователь с заданным именем.
2. Выясняет, работает ли в текущий момент пользователь с заданным именем.
3. Выводит список зарегистрированных пользователей с именами их домашних каталогов и числовыми идентификаторами, в порядке убывания идентификаторов (для редактирования вывода используйте команды **grep** или **sed**).

Задание 4. Реализуйте два исполняемых файла, выполняющих следующие задачи:

1. Расчет значения функции sin(x) с шагом 0.01 от 0 до бесконечности.
2. Расчет значения функции cos(x) с шагом 0.001 от 0 до бесконечности.
3. Для реализации исполняемых файлов можно воспользоваться компиляторами **gcc** и **g++**.
4. Запустите первый файл в фоновом режиме.
5. Запустите второй файл в приоритетном режиме.
6. Переведите второй процесс в фоновый режим.
7. Переведите первый процесс в приоритетный режим.
8. Для каждого из процессов выведите:

- долю оперативной памяти, занимаемой процессом
- время, прошедшее с момента запуска процесса
- имя управляющего терминала
- время старта процесса

4. Служебные программы ОС Windows

4.1. Теоретический блок

При установке ОС Windows вместе с ней устанавливаются Стандартные программы, позволяющие выполнять элементарные пользовательские задачи. Всем хорошо известны такие стандартные программы, как текстовый редактор Блокнот, графический редактор Paint, Калькулятор и т. п. Но кроме программ, которые прописываются в Главном меню, по умолчанию при установке ОС в каталог Windows устанавливается еще несколько различных программ. Далее перечислены некоторые из них. Описание многих из перечисленных программ имеется в справке Windows

Консольные программы

HOSTNAME.EXE

Местонахождение: Windows\system32

Утилита командной строки *hostname* выводит имя системы, на котором была запущена эта команда.

У данной команды нет параметров.

IPCONFIG.EXE

Местонахождение: Windows\system32

Описание: Конфигурация IP

Служит для отображения всех текущих параметров сети TCP/IP и обновления параметров DHCP и DNS. При вызове команды **ipconfig** без параметров выводится только IP-адрес, маска подсети и основной шлюз для каждого сетевого адаптера.

Синтаксис

ipconfig [/**all**] [/**renew** [*адаптер*]] [/**release** [*адаптер*]] [/**flushdns**] [/**displaydns**] [/**registerdns**] [/**showclassid** *адаптер*] [/**setclassid** *адаптер* [*код_класса*]]

Параметры

/all

Вывод полной конфигурации TCP/IP для всех адаптеров. Без этого параметра команда **ipconfig** выводит только IP-адреса, маску подсети и основной шлюз для каждого адаптера. Адаптеры могут представлять собой физические интерфейсы, такие как установленные сетевые адаптеры, или логические интерфейсы, такие как подключения удаленного доступа.

/renew [*адаптер*]

Обновление конфигурации DHCP для всех адаптеров (если адаптер не задан) или для заданного *адаптера*. Данный параметр доступен только на компьютерах с адаптерами, настроенными для автоматического получения IP-адресов. Чтобы указать адаптер, введите без параметров имя, выводимое командой **ipconfig**.

/release [*адаптер*]

Отправка сообщения DHCPRELEASE серверу DHCP для освобождения текущей конфигурации DHCP и удаление конфигурации IP-адресов для всех адаптеров (если адаптер не задан) или для заданного *адаптера*. Этот адаптер отключает протокол TCP/IP для адаптеров, настроенных для автоматического получения IP-адресов. Чтобы указать адаптер, введите без параметров имя, выводимое командой **ipconfig**.

/flushdns

Сброс и очистка содержимого кэша сопоставления имен DNS клиента. Во время устранения неполадок DNS эту процедуру используют для удаления из кэша записей отрицательных попыток сопоставления и других динамически добавляемых записей.

/displaydns

Отображение содержимого кэша сопоставления имен DNS клиента, включающего записи, предварительно загруженные из локального файла Hosts, а также последние полученные записи ресурсов для запросов на сопоставление имен. Эта информация используется службой DNS клиента для быстрого сопоставления часто встречаемых имен без обращения к указанным в конфигурации DNS-серверам.

/registerdns

Динамическая регистрация вручную имен DNS и IP-адресов, настроенных на компьютере. Этот параметр полезен при устранении неполадок в случае отказа в регистрации имени DNS или при выяснении причин неполадок динамического обновления между клиентом и DNS-сервером без перезагрузки клиента. Имена, зарегистрированные в DNS, определяются параметрами DNS в дополнительных свойствах протокола TCP/IP.

/showclassid *адаптер*

Отображение кода класса DHCP для указанного адаптера. Чтобы просмотреть код класса DHCP для всех адаптеров, вместо параметра *адаптер* укажите звездочку (*****). Данный параметр доступен только на компьютерах с адаптерами, настроенными для автоматического получения IP-адресов.

/setclassid *адаптер* [*код_класса*]

Задание кода класса DHCP для указанного адаптера. Чтобы задать код класса DHCP для всех адаптеров, вместо параметра *адаптер* укажите звездочку (*****). Данный параметр доступен только на компьютерах с адаптерами, настроенными для автоматического получения IP-адресов. Если код класса DHCP не задан, текущий код класса удаляется.

/?

Отображение справки в командной строке.

Примечания:

1) Команда **ipconfig** является эквивалентом для командной строки команды **winipcfg**, имеющейся в Windows Millennium Edition, Windows 98 и Windows 95. Хотя Windows XP не имеет графического эквивалента команде **winipcfg**, для просмотра и обновления IP-адреса можно воспользоваться окном «Сетевые подключения». Для этого откройте окно Сетевые подключения, щелкните правой кнопкой мыши сетевое подключение, выберите команду **Состояние**, а затем откройте вкладку **Поддержка**.

2) Данная команда доступна только на компьютерах с адаптерами, настроенными для автоматического получения IP-адресов. Это позволяет пользователям определять, какие значения конфигурации были получены с помощью DHCP, APIPA или другой конфигурации.

3) Если имя *адаптер* содержит пробелы, его следует заключать в кавычки (т. е. **"***имя_адаптера***"**).

4) В именах адаптеров, задаваемых для команды **ipconfig**, поддерживается использование подстановочного знака звездочки (*) для задания имен, начинающихся с указанной строки или содержащих указанную строку. Например, имя **Подкл*** будет включать все адаптеры, начинающиеся со строки «Подкл», а имя ***сет*** — все адаптера, содержащие строку «сет».

5) Эта команда доступна, только если в свойствах сетевого адаптера в объекте Сетевые подключения в качестве компонента установлен **протокол Интернета (TCP/IP)**.

Примеры

Чтобы вывести основную конфигурацию TCP/IP для всех адаптеров, введите: **ipconfig**

Чтобы вывести полную конфигурацию TCP/IP для всех адаптеров, введите: **ipconfig /all**

Чтобы обновить конфигурацию IP-адреса, назначенного DHCP-сервером, только для адаптера **Подключение по локальной сети**, введите: **ipconfig /renew "Подключение по локальной сети"**

Чтобы сбросить кэш сопоставления имен DNS при наличии неполадок в сопоставлении имен, введите: **ipconfig /flushdns**

Чтобы вывести код класса DHCP для всех адаптеров с именами, начинающимися со слова *Подключение*, введите: **ipconfig /showclassid Подключение***

Чтобы задать код класса DHCP *TEST* для адаптера **Подключение по локальной сети**, введите: **ipconfig /setclassid "Подключение по локальной сети" TEST**

OPENFILES.EXE

Местонахождение: Windows\system32

Запрашивает или отображает открытые файлы. Также запрашивает, отображает или разъединяет файлы, открытые сетевыми пользователями.

Подкоманда openfiles disconnect

Разъединяет одного или нескольких удаленных пользователей, присоединенных к открытым общим файлам.

Синтаксис:

openfiles.exe /disconnect [**/s** *Компьютер* [**/u** *домен\пользователь* [**/p** *пароль*]]] {[**/id** *КодОткрытогоФайла*]|[**/a** *ИмяПользователя*]|[**/o** *РежимОткрытия*]} [**/se** *ИмяСессии*] [**/op** *ИмяОткрытогоФайла*] Параметры

Параметры

/s *компьютер*

Имя или IP-адрес удаленного компьютера. Не используйте обратную косую черту. По умолчанию используется локальный компьютер.

/u *домен\пользователь*

Выполнение команды с разрешениями учетной записи пользователя, который указан как *пользователь* или *домен\пользователь*. По умолчанию используются разрешения текущего вошедшего пользователя компьютера, с которого поступила эта команда.

/p *пароль*

Определяет пароль учетной записи пользователя, заданной параметром **/u**.

/id *КодОткрытогоФайла*

Разъединяет файл, открытый со специальным цифровым параметром *КодОткрытогоФайла* на компьютере, указанном параметром **/s**. Для определения кода файла следует использовать команду **openfiles.exe /query**. Подстановочный знак (*) может быть использован для разъединения всех открытых файлов на указанном компьютере.

/a *ИмяПользователя*

Разъединяет все открытые файлы, доступ к которым осуществлялся указанным пользователем на компьютере, указанном параметром **/s**. С помощью подстановочного знака (*) можно отключить все открытые файлы на указанном компьютере.

/o *РежимОткрытия*

Разъединяет все открытые файлы с указанным параметром *РежимОткрытия* на компьютере, указанном параметром **/s**. Параметр *РежимОткрытия* включает режим чтения/записи и режим чтения. С помощью подстановочного знака (*) можно отключить все открытые файлы на указанном компьютере.

/se *ИмяСессии*

Разъединяет все открытые файлы, которые были созданы указанной сессией на компьютере, указанном параметром **/s**. С помощью подстановочного знака (*) можно отключить все открытые файлы на указанном компьютере.

/op *ИмяОткрытогоФайла*

Разъединяет открытый файл, который был создан с указанным параметром *ИмяОткрытогоФайла* на компьютере, указанном параметром **/s**. С помощью подстановочного знака (*) можно отключить все открытые файлы на указанном компьютере.

/?

Отображает справку в командной строке.

Подкоманда openfiles query

Запрашивает и отображает открытые файлы.

Синтаксис

openfiles.exe /query [**/s** *компьютер* [**/u** *домен\пользователь* [**/p** *пароль*]]] [**/fo** {**TABLE**|**LIST**|**CSV**}] [**/nh**] [**/v**] Параметры

/s *компьютер*

Имя или IP-адрес удаленного компьютера. Не используйте обратную косую черту. По умолчанию используется локальный компьютер.

/u *домен\пользователь*

Выполнение команды с разрешениями учетной записи пользователя, который указан как *пользователь* или *домен\пользователь*. По умолчанию используются разрешения текущего вошедшего пользователя компьютера, с которого поступила эта команда.

/p *пароль*

Определяет пароль учетной записи пользователя, заданной параметром **/u**.

/fo {**TABLE**|**LIST**|**CSV**}

Формат выходных данных запроса. Допустимые значения: **TABLE**, **LIST** и **CSV**. По умолчанию для выходных данных используется значение **TABLE**.

/nh

Запрещает вывод заголовка столбца. Данный параметр является допустимым, если параметр **/fo** имеет значение **TABLE** или **CSV**.

/v

Задает отображение подробных сведений о задании в выходных данных.

/?

Отображает справку в командной строке.

SFC.EXE

Местонахождение: Windows\system32

Средство проверки системных файлов позволяет администратору проверить версии всех защищенных файлов. Если при проверке системных файлов обнаруживается, что защищенный файл был изменен, то данный файл заменяется его исходной версией, которая копируется из папки *%системный_корневой_каталог%\system32\dllcache* или из папки, содержащей установочные файлы Windows. Кроме того, средство проверки системных файлов проверяет папку кэша и обновляет ее содержимое. Для

использования данного средства необходимо войти в систему с учетной записью администратора или члена группы «Администраторы». Если папка кэша повреждена или не может быть использована по другим причинам, для восстановления ее содержимого запустите программу *sfc* с параметром *scannow, scanonce* или *scanboot* (***sfc /scannow, sfc /scanonce*** или ***sfc /scanboot***).

Параметры

Sfc [/Scannow] [/Scanonce] [/Scanboot] [/Revert] [/Purgecache] [/Cachesize=x]

/Scannow

Проверить все защищенные системные файлы и заменить ошибочные версии файлов исходными версиями. Начать проверку немедленно. В процессе выполнения данной команды программе *sfc* может потребоваться доступ к установочным файлам Windows.

/Scanonce

Проверить все защищенные системные файлы при следующей перезагрузке компьютера. После перезагрузки программе *sfc* может потребоваться доступ к установочным файлам Windows. При запуске программы *sfc* с параметром */Scanonce* параметру системного реестра **SfcScan** типа DWORD присваивается значение **2**. Данный параметр находится в следующем разделе реестра:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon

/Scanboot

Проверять все защищенные системные файлы при каждой загрузке компьютера. При каждой загрузке компьютера программе *sfc* может потребоваться доступ к установочным файлам Windows. При запуске программы *sfc* с параметром */Scanboot* параметру системного реестра **SfcScan** типа DWORD присваивается значение **1**. Данный параметр находится в следующем разделе реестра:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon

/Revert

Восстановить параметры проверки, используемые по умолчанию (не проверять защищенные файлы при загрузке компьютера). Использование данной команды не изменяет размер кэша. Данный параметр аналогичен параметру ***/Enable***, который использовался в Windows 2000.

/Purgecache

Очистить файловый кэш и выполнить проверку всех защищенных системных файлов. Начать проверку немедленно. В процессе выполнения данной команды программе *sfc* может потребоваться доступ к установочным файлам Windows.

/Cachesize=*x*

Установить размер файлового кэша равным *x* мегабайтам (МБ). По умолчанию размер файлового кэша равен 50 МБ. Для фактического изменения размера кэша на диске необходимо перезагрузить компьютер и запустить программу *sfc.exe* с параметром ***/purgecache***. При запуске программы *sfc* с параметром */Cachesize* параметру системного реестра **SfcQuota** типа DWORD присваивается значение *x*. Данный параметр находится в следующем разделе реестра:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon

SHUTDOWN.EXE

Местонахождение: Windows\system32

Утилита ***Shutdown*** позволяет выключать или перегружать локальный или удаленный ПК, используя командную строку или специально созданный ярлык. Простой ее вызов без каких-либо параметров обеспечивает завершение сеанса текущего пользователя. Для использования всех возможностей этой утилиты необходимо задействовать параметры командной строки.

Синтаксис

shutdown ({-l\-s\-r\-a}) (-f) (-m (WComputer Name)) (-t xx) (-c "message") (-d(u) (p):xx:yy)

Параметры

-1

завершение сеанса текущего пользователя. При наличии параметра *-t* последний имеет приоритет, то есть завершение сеанса производится для удаленного ПК.

-s

выключить локальный ПК.

-r

перезагрузка.

-a

отмена выключения ПК, игнорируются все параметры, кроме *-1* и *ComputerName*. Данный параметр может использоваться только в тот момент, когда длится так называемый период таймаута, то есть когда программа *Shutdown* выделяет пользователю время на отмену своих действий.

-f

разрешить принудительное закрытие всех работающих приложений.

-m (WComputerName)

задать удаленный компьютер, который необходимо выключить.

-t xx

задает временную задержку до вызова процедуры выключения компьютера в секундах - *xx*. По умолчанию используются 20 с.

-c "message"

эта команда позволяет задать любое сообщение, которое будет отображаться в окне программы *Shutdown*. Максимальная длина сообщения - 127 символов. Текст сообщения необходимо заключать в кавычки

-d (u)(p):xx:yy

активирует некий специальный код выключения: ***u*** - отображение пользовательского кода, ***p*** - отображение запланированного кода, ***xx*** - задает основной код (0-255), ***yy*** – задает дополнительный код (0-65536).

/?

отображает справочную информацию по программе

SYSTEMINFO.EXE

Местонахождение: Windows\system32

Выводит на экран подробные сведения о конфигурации компьютера и операционной системы, сведения о безопасности, код продукта и параметры оборудования, такие как ОЗУ, дисковое пространство и сетевые карты.

Синтаксис

systeminfo[**.exe**] [**/s** *компьютер* [**/u** *домен\пользователь* [**/p** *пароль*]]] [**/fo** {**TABLE**|**LIST**|**CSV**}] [**/nh**]

Параметры

/s *компьютер*

Указывает имя или IP-адрес удаленного компьютера (не используйте обратную косую черту). По умолчанию используется локальный компьютер.

/u *домен\пользователь*

Выполняет команду с разрешениями учетной записи пользователя, который указан как *пользователь* или *домен\пользователь*. По умолчанию используются разрешения текущего вошедшего пользователя компьютера, с которого поступила эта команда.

/p *пароль*

Определяет пароль учетной записи пользователя, заданной параметром **/u**.

/fo {**TABLE**|**LIST**|**CSV**}

Задает формат выходных данных. Допустимые значения: **TABLE**, **LIST** и **CSV**. По умолчанию для выходных данных используется формат **TABLE**.

/nh

Запрещает вывод заголовков столбцов. Данный параметр является допустимым, если параметр **/fo** имеет значение **TABLE** или **CSV**.

/?

Отображает справку в командной строке.

TASKLIST.EXE

Местонахождение: Windows\system32

Отображает список приложений и служб с кодом процесса (PID) для всех задач, выполняющихся на локальном или удаленном компьютере.

Синтаксис

tasklist[**.exe**] [**/s** *компьютер*] [**/u** *домен\пользователь* [**/p** *пароль*]] [**/fo** {**TABLE**|**LIST**|**CSV**}] [**/nh**] [**/fi** *фильтр* [**/fi** *фильтр2* [...]]] [**/m** [*модуль*] | **/svc** | **/v**]

Параметры

/s *компьютер*

Указывает имя или IP-адрес удаленного компьютера (не используйте обратную косую черту). По умолчанию используется локальный компьютер.

/u *домен\пользователь*

Выполняет команду с разрешениями учетной записи пользователя, который указан как *пользователь* или *домен\пользователь*. По умолчанию используются разрешения текущего вошедшего пользователя компьютера, с которого поступила эта команда.

/p *пароль*

Определяет пароль учетной записи пользователя, заданной параметром **/u**.

/fo {**TABLE**|**LIST**|**CSV**}

Задает формат выходных данных. Допустимые значения: **TABLE**, **LIST** и **CSV**. По умолчанию для выходных данных используется формат **TABLE**.

/nh

Запрещает вывод заголовков столбцов. Данный параметр является допустимым, если параметр **/fo** имеет значение **TABLE** или **CSV**.

/fi *имя_фильтра*

Задает типы процессов, которые следует завершить или не следует. Допустимыми именами фильтров, операторами и значениями являются имена, приведенные в таблице 4.1.

/m [*модуль*]

Задает вывод сведений о модулях для каждого процесса. При указании модуля отображаются все процессы, использующие этот модуль. Если модуль не определен, выводятся на экран все процессы для всех модулей. Нельзя использовать совместно с параметрами ***/svc*** и ***/v***

/svc

Отображает без обрезки сведения о всех службах для каждого процесса. Данный параметр является допустимым, если параметр ***/fo*** имеет значение ***TABLE***. Нельзя использовать совместно с параметрами ***/m*** и ***/v***

/v

Задает отображение подробных сведений о задании в выходных данных. Нельзя использовать совместно с параметрами ***/svc*** и ***/m***.

/?

Отображает справку в командной строке.

Примечания

Команда **tasklist** является заменой средству **TList**

Таблица 4.1 – Допустимые значения фильтра команды TASKLIST.EXE

Имя	Операторы	Значение
Status	eq, ne	RUNNING|NOT RESPONDING
Imagename	eq, ne	Любая допустимая строка
PID	eg, ne, gt, lt, ge, le	Любой положительное число
Session	eg, ne, gt, lt, ge, le	Любой действительный номер сеанса.
SessionName	eq, ne	Любая допустимая строка
CPUTime	eg, ne, gt, lt, ge, le	Допустимое время в формате *чч:мм:сс*. Компоненты *мм* и *сс* должны иметь значения от 0 до 59, а *чч* может быть любым значением числа без знака
Memusage	eg, ne, gt, lt, ge, le	Любое целое число
Username	eq, ne	Любое действительное имя пользователя ([*домен*\]*пользователь*)
Services	eq, ne	Любая допустимая строка
Windowtitle	eq, ne	Любая допустимая строка
Modules	eq, ne	Любая допустимая строка

TASKKILL.EXE

Местонахождение Windows\system32

Завершает одно или несколько заданий или процессов. Процессы могут быть уничтожены кодом процесса или именем образа.

Синтаксис

taskkill [**/s** *компьютер*] [**/u** *домен\пользователь* [**/p** *пароль*]]] [**/fi** *имя_фильтра*] [**/pid** *код_процесса*]|[**/im** *имя_образа*] [**/f**][**/t**]

Параметры

/s *компьютер*

Указывает имя или IP-адрес удаленного компьютера (не используйте обратную косую черту). По умолчанию используется локальный компьютер.

/u *домен\пользователь*

Выполнение команды с разрешениями учетной записи пользователя, который указан как *пользователь* или *домен\пользователь*. По умолчанию используются разрешения текущего вошедшего пользователя компьютера, с которого поступила эта команда.

/p *пароль*

Определяет пароль учетной записи пользователя, заданной параметром **/u**.

/fi *имя_фильтра*

Задает типы процессов, которые следует завершить и не следует. Допустимыме имена фильтров, операторов и значений приведены в таблице 4.2.

Таблица 4.2 – Допустимые значения фильтра команды TASKKILL.EXE

Имя	Операторы	Значение
Hostname	eq, ne	Любая допустимая строка
Status	eq, ne	RUNNING|NOT RESPONDING
Imagename	eq, ne	Любая допустимая строка
PID	eg, ne, gt, lt, ge, le	Любой положительное число
Session	eg, ne, gt, lt, ge, le	Любой действительный номер сеанса
CPUTime	eg, ne, gt, lt, ge, le	Допустимое время в формате *чч*:*мм*:*сс*. Компоненты *мм* и *сс* должны иметь значения от 0 до 59, а *чч* может быть любым значением числа без знака
Memusage	eg, ne, gt, lt, ge, le	Любое целое число
Username	eq, ne	Любое действительное имя пользователя ([*домен*\]*пользователь*).
Services	eq, ne	Любая допустимая строка
Windowtitle	eq, ne	Любая допустимая строка

/pid *код_процесса*

Указывает код процесса, который необходимо завершить.

/im *имя_образа*

Указывает имя образа процесса, который необходимо завершить. Используйте подстановочный знак (*****) для указания всех имен образа.

/f

Указывает, что процесс(ы) должен быть принудительно завершен. Этот параметр не действует для удаленных процессов, все удаленные процессы завершаются принудительно.

/t

Задает завершение всех дочерних процессов вместе с родительским, такое действие обычно известно как уничтожение дерева.

/?

Отображает справку в командной строке.

Примечания:

1) Подстановочный символ (*) принимается только при указании вместе с фильтрами.

2) Завершение удаленных процессов всегда выполняется принудительно независимо от указания параметра **/f**.

3) Указание имени компьютера в качестве фильтра HOSTNAME приведет к завершению работы и остановке всех процессов.

4) Используйте команду **tasklist** для определения кода завершаемого процесса.

5) Команда **taskkill** является заменой средству **Kill**.

MSINFO32.EXE

Местонахождение: C:\Program Files\Common Files\Microsoft Shared\MSInfo

Служит для отображения подробных сведений об оборудовании, системных компонентах и среде программного обеспечения.

Синтаксис

msinfo32 [**/?**] [**/pch**] [**/nfo** *имя_файла*] [**/report** *имя_файла*] [**/computer** *имя_компьютера*] [**/showcategories**] [**/category** *код_категории*] [**/categories** *код_категории*]

Параметры

имя_файла

Файл, который требуется открыть. Файл может иметь расширение NFO, XML, TXT или CAB.

/?

Отображение справки по команде *msinfo32*.

/pch

Отображение журнала.

/nfo *имя_файла*

Сохранение экспортированного файла как NFO-файла.

/report *имя_файла*

Сохранение экспортированного файла как TXT-файла.

/computer *имя_компьютера*

Открытие окна сведений о системе для указанного удаленного компьютера.

/showcategories

Открытие окна сведений о системе, содержащего все доступные коды категорий.

/category *код_категории*

Открытие окна сведений о системе, в котором выбрана указанная категория. Для отображения списка доступных кодов категорий служит параметр **/showcategories**

/categories *код_категории*

Открытие окна сведений о системе, содержащего только указанные категории. Вывод также ограничивается только выбранными категориями. Для отображения списка доступных кодов категорий служит параметр **/showcategories**

Заметки

Некоторые категории сведений о системе содержат большие объемы данных. Скорость создания отчетов для этих категорий можно увеличить, используя команду ***start /wait***.

Примеры

Чтобы получить список доступных кодов категорий, введите: ***msinfo32 /showcategories***

Чтобы открыть окно сведений о системе, содержащее все доступные сведения, кроме сведений о загруженных модулях, введите: ***msinfo32 /categories +all -loadedmodules***

Чтобы открыть окно сведений о системе и создать NFO-файл syssum.nfo, содержащий сведения категории «Сведения о системе», введите: ***msinfo32 /nfo syssum.nfo /categories +systemsummary***

Чтобы вывести сведения о конфликте ресурсов и создать NFO-файл conflicts.nfo, содержащий сведения о конфликтах ресурсов, введите: ***msinfo32 /nfo conflicts.nfo /categories +componentsproblemdevices+ resourcesconflicts+resourcesforcedhardware***

Оконные программы

CHARMAP.EXE

Местонахождение: Windows\system32

Таблица символов служит для просмотра символов, включенных в выбранный шрифт. Она отображает следующие наборы символов: Windows, DOS и Юникод.

Отдельный символ или группу символов можно скопировать в буфер обмена, а затем вставить в любое приложение, в котором они будут отображаться. Многие программы, например WordPad, позволяют копировать символы путем их перетаскивания из таблицы символов непосредственно в открытый документ. Более подробное описание программы есть в справке.

TASKMGR.EXE

Местонахождение: Windows\system32.

Запустить программу можно как по имени файла taskmgr.exe, так и одновременным нажатием клавиш Ctrl+Alt+Del или Ctrl+Shift+Esc.

CLEANMGR.EXE

Местонахождение: WINDOWS\system32

Программа очистки диска используется для освобождения пространства на жестком диске с помощью удаления временных файлов Интернета, установленных компонентов и программ, которые больше не используются, и очистки корзины. Программу можно запустить через *Пуск - Все программы - Стандартные - Служебные - Очистка диска* или через *Пуск - Выполнить - cleanmgr*

CIADV.MSC

Местонахождение: Windows\system32

Служба индексирования Windows. Служба индексирования извлекает сведения из набора документов и собирает их в структуру, обеспечивающую быстрый и удобный доступ к этим сведениям с помощью команды ***Найти*** Windows, формы запроса службы индексирования или обозревателя. Эти сведения могут включать текст (содержимое) документа, характеристики и параметры (свойства) документа, такие как имя автора. После создания индекса можно запросить в нем документы, содержащие ключевые слова, фразы или свойства. Например, можно запросить все документы, содержащие слово «продукт», либо все документы Microsoft Office, созданные определенным автором.

Служба индексирования возвращает список всех документов, соответствующих указанным условиям поиска.

DEVMGMT.MSC

Местонахождение: Windows\system32

Диспетчер устройств используют для обновления драйверов (или программного обеспечения) оборудования, изменения настроек оборудования, а также для устранения неполадок.

DISKMGMT.MSC

Местонахождение: Windows\system32

Служба ***Управление дисками*** предназначена для таких задач, как создание и форматирование разделов и томов и назначение букв дисков

EVENTVWR.MSC

Местонахождение: Windows\system32

В окне просмотра событий ведутся журналы программных, системных событий, а также событий безопасности на компьютере. Окно просмотра событий используется для просмотра журналов событий и управления ими, получения сведений о неполадках аппаратного и программного обеспечения, а также для наблюдения за событиями безопасности Windows.

LUSRMGR.MSC

Местонахождение: Windows\system32

Описание: Локальные пользователи и группы - Local Users and Groups.

Служба ***Локальные пользователи и группы —*** это инструмент, предназначенный для управления локальными пользователями и группами.

SERVICES.MSC

Местонахождение: Windows\system32

Программа «Службы» используется для управления службами на локальном или удаленном компьютерах.

PERFMON.MSC

Местонахождение: Windows\system32

Описание: Производительность (Performance Monitor)

Огромное количество счетчиков производительности компьютера (в том числе компьютера в сети), загруженность процессора, памяти, кэша, логических дисков. Большое количество настроек и возможностей. Можно, например, узнать активность жесткого диска, его загруженность или отследить, сколько байт занимает отдельно взятая программа в файле подкачки и т. д.

4.2. Практический блок

Внимание! После завершения выполнения всех заданий необходимо будет вернуть систему в исходное состояние. Рекомендуется перед началом работы записать исходные настройки системы.

Задание 1. *Работа с настройками TCP/IP*

Создайте пакетный файл, который, в зависимости от выбора пользователя выполняет следующие действия:

1) выводит имя компьютера;
2) выводит полную конфигурацию TCP/IP для всех адаптеров;
3) отображает содержимое кэша сопоставления имен DNS клиента;
4) очищает кэш DNS.

Задание 2. *Утилита SHUTDOWN.EXE*

1. Пошлите сигнал перезагрузки с таймаутом 60 сек. и сообщением "Тестовая перезагрузка".
2. Отмените перезагрузку.

Задание 3. *Работа с процессами*

1. Создайте при помощи любого доступного компилятора исполняемый файл, содержащий бесконечный цикл.
2. Запустите файл на выполнение.
3. Выясните PID зациклившейся программы и количество памяти занимаемое ей.
4. Принудительно завершите программу.

Задание 4. *Управление компьютером*

1. Запустите Управление компьютером, с помощью справочной системы ознакомьтесь с возможностями этой программы.
2. Выполните просмотр журнала событий.
3. Ознакомьтесь со списком локальных пользователей.
4. Получите сведения об отказах в системе.
5. Создайте журнал счетчиков производительности системы.
6. Проверьте все системные файлы.
7. Создайте текстовый файл, содержащий сведения о системе.
8. Создайте пользователя при помощи LUSRMGR.MSC.
9. Создайте группу, добавьте пользователя в группу.

5. Реестр Windows

5.1. Теоретический блок

Реестр - это база данных для хранения сведений о конфигурации компьютера и настроек операционной системы. Реестр содержит данные, к которым Windows постоянно обращается во время загрузки, работы и ее завершения, а именно:

- настройки для всех пользовательских профилей;
- сведения о конфигурации установленного оборудования;
- данные об установленных программах и типах документов, создаваемых каждой программой;
- свойства папок и значков программ;
- данные об используемых портах и т. п.

Без нормального состояния реестра система просто не загрузится. Любое повреждение реестра вызывает множество проблем, даже если система загружается. При работе с реестром следует поступать очень осторожно. Необходимо выполнять резервное копирование, прежде чем что-либо изменять в реестре.

Внутренняя организация реестра

Для работы с реестром используется редактор **Regedit**, который можно вызвать по команде Главного меню - ***Выполнить...*** В редакторе реестра есть две панели. Элементы, отображаемые в левой панели, являются ключами или разделами, а в правой панели - значениями. Значение - это содержимое реестра, подобно абзацам в книге.

Раздел реестра отображается как файл на жестком диске. Под кустом понимают набор разделов, подчиненных разделов и их параметров.

В Windows 7 реестр содержит следующие основные разделы:

HKEY_CLASSES_ROOT (HKCR) - ссылка на раздел HKEY_LOCAL_MACHINE\Software\Classes. Хранящиеся здесь сведения обеспечивают запуск необходимой программы при открытии файла с помощью проводника. Этот раздел содержит связи между приложениями и типами файлов, а также информацию об OLE.

HKEY_CURRENT_USER (HKCU) - ссылка на определенный подраздел HKEY_USERS. Настройки соответствуют текущему, активному пользователю, выполнившему вход в систему.

HKEY_LOCAL_MACHINE (HKLM) - раздел содержит настройки, относящиеся к текущему компьютеру и действительные для всех пользователей, а также информацию об аппаратной конфигурации и установленном программном обеспечении.

HKEY_USERS (HKU) - этот раздел содержит настройки для всех пользователей компьютера.

HKEY_CURRENT_CONFIG (HKCC) - ссылка на HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Hardware Profiles\Current. Раздел содержит сведения о настройках оборудования, используемых локальным компьютером при запуске системы, т. е. информацию о текущей конфигурации.

Файлы реестра (имени файла соответствует куст реестра):

- SAM - HKEY_LOCAL_MACH£NE\SAM;
- SECURITY - HKEY_LOCAL_MACHINE\Security;
- Software - HKEY_LOCAL_MACHINE\Software;
- System - HKEY LOCAL_MACHINE\System;
- HKEYCURRENTCONFIG;
- Default - HKEYUSERSVDEFAULT;
- Файлы Ntuser.dat - HKEY CURRENT USER (эти файлы хранятся в C:\Documents and Settings\%UserName%) и содержат конфигурацию для конкретного пользователя. По умолчанию почти все файлы кустов: Default, SAM, Security, Software и System, сохраняются в папке %SystemRoot%\System32\Config.
- Папка %SystemRoot%\Profiles содержит настройки для каждого пользователя компьютера. Точный список файлов реестра можно посмотреть по ветви: HKEY_LOCAL_MACHINE\System\CotnrolSet\Control\HiveList\.

При начальной загрузке к этому разделу обращается Configuration Manager, чтобы проинициализировать все основные разделы реестра.

Описание типов данных реестра

REG_BINARY - двоичные данные. Большинство сведений об аппаратных компонентах хранится в виде двоичных данных и выводится в редакторе реестра в шестнадцатеричном формате.

REG_DWORD - целые числа размером в 4 байта. Многие параметры служб и драйверов устройств имеют этот тип и отображаются в двоичном, шестнадцатеричном или десятичном форматах.

REG_EXPAND_SZ - строка данных переменной длины.

REG_MULTI_SZ - многострочный текст. Этот тип, как правило, имеют списки и другие записи в формате, удобном для чтения. Записи разделяются пробелами, запятыми или другими символами.

REG_SZ - текстовая строка фиксированной длины.

REG_FULL_RESOURCE_DESCRIPTOR – последовательность вложенных массивов, разработанная для хранения списка аппаратных ресурсов или драйверов.

REG_NONE - Параметры данного типа не имеют определенного двоичного или строкового типа. В редакторах реестра они отображаются в виде параметров двоичного типа.

REG_LINK - строковый тип данных для указания пути к файлам.

Существует ряд других типов данных

Создание текстовой копии реестра

Простой способ зарезервировать реестр заключается в создании его текстовой копии с помощью команды меню **Файл-Экспорт…** редактора **REGEDIT.** Рекомендуется регулярно выполнять такую операцию. Для восстановления копии реестра следует выполнить команду - **Файл-Импорт…**

Основные стандартные разделы нельзя удалить или переименовать. Некоторые разделы реестра являются энергозависимыми и не хранятся в каком-либо файле. Операционная система создает и управляет этими разделами полностью в памяти. Система создает энергозависимые разделы каждый раз при начальной загрузке. Например, HKEY_LOCAL_MACHINE \HARDWARE - раздел реестра, который хранит информацию о физических устройствах и назначенным им ресурсам. Назначение ресурса и аппаратное обнаружение происходят каждый раз при загрузке системы, поэтому логично, что эти данные не записываются на диск.

Описание основных разделов

Основной раздел системного реестра - это раздел HKEY_LOCALMACHINE \SYSTEM.

Наибольший интерес для пользователей представляют ветви HKEY_CURRENT_USER и HKEY_LOCAL_MACHINE: именно там хранятся настройки, изменение которых можно выполнять в соответствии с эргономическими требованиями. Раздел HKEY_USERS содержит все активные загруженные параметры пользователя. Он имеет не менее трех ключей:

1. Подраздел DEFAULT, где хранится используемая конфигурация, когда ни один из пользователей еще не вошел в компьютер. То есть отображается приглашение на вход в систему.

2. Дополнительный подраздел, который имеет имя в соответствии с security ID (SID) текущего пользователя. Этот подключ реестра содержит конфигурацию текущего пользователя. Если пользователь вошел удаленно, данные для конфигурации пользователя сохраняются в системном реестре местного компьютера. Данные из HKEY_USERS\%SID% также появляются в HKEY_CURRENT_USER.

3. Дополнительный подраздел, который имеет имя в соответствии с SID текущего пользователя с суффиксом Classes. Этот раздел содержит классы текущего пользователя. Данные в HKEY_USERS\%SID%_Classes также содержатся в HKEY_CLASSES_ROOT.

В Windows'XP конфигурация пользователя по умолчанию (default user profile) не хранится в системном реестре. Она находится на системном диске в файле \Documents and Settings\DefaultUser\Ntuser.dat.

Куст HKEY_CURRENT_USER является ссылкой на определенный подраздел куста HKEY_USERS. Это значит, что все изменения в разделах,

подразделах и ключах куста HKEY_CURRENT_USER автоматически тут же отображаются в определенном подразделе HKEY_USERS, соответствующем активному пользователю, т. е. пользователю, выполнившему процедуру входа.

Чтобы узнать, в каком именно разделе HKEY_USERS проводятся изменения, пользователь должен узнать свой SID. Тогда искомый раздел будет именоваться примерно следующим образом: HKEY USERS\S-1-5-21-117609710-1606980848-839522115-500. Где цифровая часть, вместе с буквой S, и есть SID.

SID – идентификатор безопасности структура данных переменной длины, которая идентифицирует пользователя, группу или компьютер. Каждая учетная запись в сети имеет уникальный SID. Внутренние процессы в Windows обращаются к SID для получения учетной записи пользователя или имени группы. Для выяснения имени пользователя выполните следующие действия:

- откройте редактор реестра HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList;

- выберите SID и посмотрите ProfilelmagePath, в конце строки отображается имя пользователя.

Увидеть все SID, относящиеся к пользователям компьютера, можно в разделе HKEY_USERS. Но в большинстве случаев знать SID не обязательно.

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet - это тоже ссылка на один из пронумерованных подразделов с именами HKEY_LOCAL_MACHINE\SYSTEM\ControlSet00n, где n - номер. Подразделы ControlSet00n представляют собой наборы настроек для операционной системы. Большинство систем имеют два пронумерованных управляющих набора: оригинал и резервную копию, которая использовалась при последнем успешном старте системы. Операционная система может обслуживать несколько таких наборов. Эти значения сохранены в подключе Select. Имена параметров ключа Select соответственно: Current — текущие настройки и Default - настройки по умолчанию. Параметр LastKnownGood соответствует пункту меню при загрузке «Загрузка последней удачной конфигурации».

REG-файлы

REG-файл - это файл с расширением *.reg*, имеющий определенную структуру и содержащий информацию, которая импортируется в реестр. Создать такой файл можно в любом текстовом редакторе (например, Блокнот). Перенос информации из такого файла в реестр производится путем запуска этого файла.

Reg-файлы могут пригодиться в различных ситуациях. Например, если была заблокирована работа с редактором реестра, то наиболее простым спо-

собом исправить установки в реестре будет создание и импортирование reg-файла.

Первая строка reg-файла для Windows обязательно должна содержать:

REGEDIT4

или

Windows Registry Editor Version 5.00

Регистр символов важен!

Вторая строка обязательно должна быть ***пустой!*** Затем указывается раздел реестра, в котором надо прописать или изменить какие-то параметры. Название раздела должно быть заключено в квадратные скобки. Затем ниже прописываются параметры и значения по одному параметру в строке. При окончании описания параметров и их значений обязательно оставляют ***пустую последнюю строку!***

В общем виде структура reg-файла выглядит следующим образом:

REGEDIT4

[Razdell]
" param 1 "=" znachenie 1"
" param2 "=" znachenei2 "
"pazam3 "="znachenie3 "

[Razdel2]
"param_l "="znachenie _ 1 "

Для параметров типа DWORD используется строка ***"param"=dword***:*XXXXXXXX* "param" - имя параметра, dword - указывает на тип этого параметра (буквы должны быть обязательно маленькие) и после двоеточия следует значение из восьми цифр в шестнадцатеричном формате. Однако большинство параметров DWORD имеют значение либо 0, либо 1, значит, нужно написать соответственно либо 00000000, либо 00000001 вместо значков XXXXXXXX. Пробелы в строке не допускаются.

Для добавления двоичного параметра формат записи несколько иной:

"param "=hex:XX,XX,*XX,....*

"param" - имя параметра, после знака "=" идет hex, т. е. указывается, что это будет двоичный параметр, затем идут шестнадцатеричные числа, отделенные запятой. Например, если вам надо добавить двоичный параметр равный "be 00 00 00", то следует записать строку: ***"param"=hex:be,00,00,00***

В реестре существуют параметры "По умолчанию" ("Default"). Чтобы присвоить им какое-то значение через reg-файл, надо добавить такую строку:

@="znachenie"

Здесь значок @ показывает, что у нас присваивается значение параметра "По умолчанию". Обратите внимание на то, что он не заключается в кавычки.

Пример reg-файла для Windows'XP:

Windows Registry Editor Version 5.00

;Отключить перезагрузку в случае BSOD
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\CrashControl]
"AutoReboot"=dword:00000000

;Отключить уведомление на экране приветствия о непрочитанных сообщениях
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\UnreadMail]
"MessageExpiryDays"=dword:00000000

Можно удалять разделы реестра и значения в разделах. Для удаления раздела используют тот же самый формат reg-файла, только перед наименованием раздела ставят знак "минус" ("-").

Например:

REGEDIT4

[-HKEY_LOCAL_MACHINE\SYSTEM\Setup]

Этот же формат используют для удаления отдельных параметров в выбранных разделах, поставив знак "минус" ("-") после знака "равно" ("="):

REGEDIT4

[HKEY_LOCAL_MACHINE\SYSTEM\Setup]
"SetupType"=-dword:0

Настройка компьютера

Настройка рабочего стола, панели задач и значков

Выбор обоев для рабочего стола

Картинке, которая используется в качестве обоев для рабочего стола, соответствует параметр *Wallpaper* из раздела HKCU\Control Panel\Desktop. Он является строковым (REG_SZ) и указывает путь к BMP-файлу, который должен использоваться в качестве обоев рабочего стола. Например, если картинка (Wallpaper.bmp) находится в корневой директории, то в качестве строкового параметра необходимо указать C:\Wallpaper.bmp.

Выбирать обои путем редактирования реестра нецелесообразно, проще сделать это стандартными средствами Windows Vista. Однако с их помощью

нельзя запретить пользователю изменять обои рабочего стола. Такой запрет можно реализовать как раз путем редактирования реестра. Для запрета изменения обоев рабочего стола в разделе HKCU\Software\Microsoft\Windows\CurrentVersion\Policies нужно создать новый раздел (ключ) System, а в нем — параметр *NoDispBackgroundPage* типа REG_DWORD, значение которого устанавливается равным. Изменения начнут действовать после перезагрузки компьютера. Чтобы вернуть все в исходное состояние, следует присвоить параметру NoDispBackgroundPage значение 0.

Отображение версии Windows на рабочем столе

Для того чтобы в правом нижнем углу рабочего стола отображался номер сборки Windows (например, Windows Vista Build 6000), в разделе реестра HKCU\Control Panel\Desktop необходимо создать (если его еще нет) параметр *PaintDesktopVersion* типа REG_DWORD, которому присваивается значение 1. Результат можно наблюдать после перезагрузки компьютера. Для запрета отображения номера сборки Windows на рабочем столе параметру PaintDesktopVersion присваивается значение 0.

Сокрытие элементов на рабочем столе

Скрыть все элементы (иконки) на рабочем столе можно путем создания в разделе HKCU\Software\Microsoft\Windows\CurrentVersion\ Policies раздела Explorer, в котором необходимо создать параметр *NoDesktop* типа REG_DWORD, имеющий значение 1. Чтобы вернуть все в исходное состояние, достаточно присвоить параметру NoDesktop значение 0.

Добавление значка корзины в папку «Мой компьютер»

Чтобы добавить значок корзины в окно «Мой компьютер», необходимо в раздел реестра HKLM\Software\Microsoft\Windows\ CurrentVersion\ Explorer\MyComputer\NameSpace добавить подраздел {645FF040-5081-101B-9F08-00AA002F954E}.

Добавление команд в контекстное меню значка My Computer

Если на значке *My Computer* (Мой компьютер) щелкнуть правой кнопкой мыши, то появится контекстное меню элемента *My Computer*, в которое можно добавить собственную команду. Для этого в разделе реестра HKCR\CLSID\{20D04FE0-3AEA-1069-A2D8-08002B30309D}\shell
необходимо создать новый раздел (он и будет отображаться в меню — например *Calculator* (Калькулятор)), в котором, в свою очередь, создается подраздел *command*. Параметр по умолчанию *(Default)* данного раздела типа REG_SZ должен задавать строку для запуска приложения — например, если необходимо запустить приложение *Calculator*, параметру присваивают значение calc.exe

Удаление стрелок с ярлыков приложений на рабочем столе

Для любого файла, документа и приложения на рабочем столе можно создать ярлык — значок, который располагается в доступном месте и служит для открытия соответствующего ему файла.

На значках, соответствующих ярлыкам файлов, присутствует стрелка в нижнем левом углу, которую при желании можно удалить. Для этого в разделе реестра HKCR\lnkfile нужно удалить строковый параметр *IsShortcut*.

Настройка Главного меню

Скрытие пунктов в Главном меню: ветвь
[HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\ Current Version\Policies\Explorer]

NoClose=hex:01,00,00,00 ; завершение работы
NoFavoritesMenu=hex:01,00,00,00; избранное
NoFind=hex:01,00,00,00 ; поиск
NoLogOff=hex:01,00,00,00 ; log off
NoRecentDocsMenu=hex:01,00,00,00 ; документы
NoRun=hex:01,00,00,00 ; выполнить
NoSefFolders=hex:01,00,00,00 ; настройки
Для возврата пунктов после hex : вместо 01 указывайте 00

Управление дисками

Если вы хотите скрыть значки дисков в окне Мой компьютер и Проводник, то откройте раздел HKCU\Software\Microsoft\Windows\ CurrentVersion\Policies\Explorer и создайте параметр NoDrives типа **DWORD** с требуемым значением. Также будут скрыты эти значки и в стандартных окнах Открытия и Сохранения файлов. Тем не менее, пользователь по-прежнему имеет доступ к этим дискам (через команду Выполнить или печатая вручную адрес в адресной строке Проводника)

Данный параметр является набором битовых флагов. Каждый бит соответствует одному из 26 возможных имен дисков. Каждому диску присваиваются значения (hex): A -1; B - 2; C - 4 и т.д. Чтобы скрыть нужные вам диски, нужно сложить эти биты. Сложность состоит в переводе двоичного значения в шестнадцатиричное. В таблице 5.1 приводится небольшой список этих значений.

Можно не скрывать сами значки дисков, но запретить пользователю доступ к файлам заданных дисков через Проводник, Мой компьютер, Выполнить или команду Dir. Откройте реестр и создайте параметр NoViewOnDrive типа **DWORD** в разделе HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer, содержащий битовую маску для дисков. Например, диск A имеет бит 1, диск C - 4, диск D - 8. Таким образом, чтобы скрыть диски A и D, нужно сложить

их значения 1 (A) + 8 (D) и установить значение 9. Список всех дисков:

A: 1, B: 2, C: 4, D: 8, E: 16, F: 32, G: 64, H: 128, I: 256, J: 512, K: 1024, L: 2048, M: 4096, N: 8192, O: 16384, P: 32768, Q: 65536, R: 131072, S: 262144, T: 524288, U: 1048576, V: 2097152, W: 4194304, X: 8388608, Y: 16777216, Z: 33554432, Все диски: 67108863

Таблица 5.1 – Возможные значения параметра NoDrives

0x03FFFFFF	Скрывает все значки
0x3	Скрывает только диски A и B
0x4	Скрывает только диск C
0x8	Скрывает только диск D
0x7	Скрывает только диски A, B и C
0xF	Скрывает только диски A, B, C и D
0x0	Видны все диски

Настройка загрузки Windows

Автоматический вход в систему

Существует возможность автоматического входа в Windows, минуя экран приветствия, когда приходится указывать имя пользователя и пароль.

Для автоматического входа в систему в разделе реестра HKLM\SOFTWARE\Microsoft\ Windows NT\CurrentVersion\Winlogon требуется изменить значение параметра *AutoAdminLogon* типа REG_DWORD, установив его равным 1 (аналогичная процедура в Windows XP производится в другом разделе реестра — HKLM\SOFTWARE\Microsoft\ Windows\CurrentVersion\Winlogon.

Кроме того, в том же разделе реестра необходимо задать значения строковых параметров *DefaultUserName* и *DefaultPassword* — первое из них соответствует имени пользователя, а второе задает пароль пользователя.

Если компьютер входит в состав домена, то придется задать также имя сетевого домена в строковом параметре *DefaultDomainName*.

При автоматическом входе в систему любой пользователь, получивший доступ к компьютеру, может узнать пароль, хранящийся в реестре в открытом виде.

Ограничение на число попыток автоматического входа в систему

При настройке автоматического входа в систему можно задать количество таких входов в систему.

Для этого в разделе реестра HKLM\SOFTWARE\ Microsoft\Windows NT\CurrentVersion\Winlogon следует создать параметр *AutoLogonCount* типа REG_DWORD и присвоить ему численное значение, которое будет определять количество разрешенных автоматических входов в систему. К примеру, если данный параметр равен 10, то будет разрешено 10 раз

воспользоваться этой возможностью, причем после каждого входа данный параметр будет автоматически уменьшаться на единицу. Когда значение параметра достигнет 0, ключи AutoLogonCount и DefaultPassword будут удалены из реестра, а параметру *AutoAdminLogon* будет присвоено значение 0.

Замечение. Если значение *DefaultPassword* не задано, система автоматически изменяет значение раздела реестра *AutoAdminLogon* с 1 (истина) на 0 (ложь), тем самым отключая автоматический вход.

Если во время загрузки системы удерживать нажатой клавишу Shift, то автоматический вход прервется и появится окно выбора пользователя.

Настройка сообщения при загрузке Windows

С помощью реестра можно настроить систему таким образом, чтобы при загрузке появлялось диалоговое окно с каким-либо сообщением. Для этого необходимо перейти в раздел реестра HKLM\SOFTWARE\Microsoft \WindowsNT\CurrentVersion\Winlogon и создать в нем строковые (REG_SZ) параметры *LegalNoticeCaption* и *LegalNoticeText*.

В качестве значения параметра LegalNoticeCaption нужно указать строку, которая будет отображаться в заголовке сообщения, а параметра LegalNoticeText — текст сообщения.

Настройка раскладки клавиатуры в окне приветствия

В большинстве случаев при работе на компьютере устанавливается английская и русская раскладки клавиатуры, при этом одна из них является основной (используемой по умолчанию), а другая — дополнительной.

Раскладки клавиатуры, доступные в окне входа в систему, задаются только на этапе установки операционной системы. И если в качестве основной была выбрана русская раскладка, а пароль пользователь предпочитает набирать на английской, то придется постоянно переключать раскладку клавиатуры в окне входа в систему. Чтобы избежать этого, необходимо в разделе реестра HKU\.Default\Keyboard Layout\Preload отредактировать строковые параметры «1» и «2». Параметр «1» соответствует раскладке клавиатуры по умолчанию, а «2» — дополнительной. Эти параметры могут иметь значения 00000409 (английская раскладка) или 00000419 (русская раскладка).

Настройка режимов работы Windows

Изменение регистрационных данных

Для того чтобы изменить регистрационные данные о праве использования программ, установленных на компьютере, необходимо в разделе HKLM\SOFTWARE\Microsoft\Windows NT\ CurrentVersion изменить строковые параметры *RegisteredOwner* и *RegisteredOrganization*. Значение

параметра RegisteredOwner задает имя владельца, а параметра RegisteredOrganization — название организации.

Настройка параметров восстановления системы

В операционной системе Windows реализована технология восстановления системы (System Restore) — для ее реализации в системе создаются точки отката, к которым впоследствии можно вернуться.

Все относящиеся к восстановлению системы параметры типа REG_DWORD находятся в разделе реестра HKLM\Software\Microsoft\ Windows NT\CurrentVersion\SystemRestore. Используются следующие параметры:

– *DiskPersent* (раздел HKLM\SOFTWARE\ Microsoft\Windows NT\CurrentVersion\SystemRestore\cfg) — в качестве его значения указывается, какой объем дискового пространства будет использовать программа. По умолчанию для дисков с объемом более 4 Гбайт используется 15%;

– *RPGlobalInterval* — параметр задает интервал времени в секундах, в течение которого длится ожидание перед созданием новой контрольной точки. По умолчанию значение составляет 86 400 с (24 ч);

– *RPLifeInterval* — параметр задает интервал времени в секундах, в течение которого программа хранит точки восстановления перед их удалением. По умолчанию — 90 дней (7 776 000 с);

– *RPSessionInterval* — параметр задает промежуток времени в секундах, в течение которого программа ожидает перед созданием новой контрольной точки при включенном компьютере. Значение 0 соответствует отключению ожидания. Можно установить свое значение (например, интервал в час), чтобы программа создавала контрольные точки через заданный интервал.

Настройка автозагрузки программ

Все параметры автозагрузки программ содержатся в разделе реестра HKLM\Software\ Microsoft\Windows\CurrentVersion, внутри которого имеются подразделы Run, RunOnce с включенными в них строковыми параметрами, отвечающими за запуск программ. Название параметра может быть произвольным (обычно оно совпадает с названием программы), а в качестве значения в этих параметрах указывается путь к запускаемой программе.

В разделе RunOnce прописываются программы, которые запускаются всего один раз. К примеру, при установке новых программ некоторые из них прописывают туда ключи, указывающие на какие-либо настроечные модули, которые запускаются сразу после перезагрузки компьютера. Такие ключи после своего запуска автоматически удаляются. Для удаления той или иной программы из автозапуска достаточно удалить соответствующий параметр из раздела Run.

К примеру, в разделе Run может находиться параметр *ICQ Lite* со значением C:\Program Files\ICQLite\ICQLite.exe — minimize, отвечающий за автоматический запуск ICQ при старте компьютера. Если вы не хотите, что бы ICQ стартовала автоматически, то данный параметр можно удалить.

Настройка контекстного меню папок и файлов

При установке программ на ПК контекстное меню файлов и папок, возникающее при щелчке правой кнопкой мыши на этих элементах, постепенно заполняется лишними пунктами. Откорректировать список всех пунктов контекстного меню можно в разделе реестра HKCR*\shellex\ContextMenuHandlers, в котором в качестве подразделов служат команды, отображаемые в меню любого файла. При необходимости ненужные команды можно удалить.

Отключение стандартного автозапуска компакт-дисков

Чтобы отключить автозапуск компакт-диска, необходимо в разделе реестра HKLM\SYSTEM\ CurrentControlSet\Services\CDRom установить значение параметра *AutoRun* типа REG_DWORD равным 0.

Кроме вышеназванного способа, существует и другой, новый метод автозапуска компакт-дисков, для отключения которого необходимо перейти в раздел реестра HKLM\SOFTWARE\ Microsoft\Windows\CurrentVersion\Explorer\ AutoplayHandlers\CancelAutoplay\Files. В нем находятся текстовые параметры, содержащие имена файлов, и если такие файлы будут присутствовать на компакт-диске, встроенный AutoRun запускаться не станет. Добавление строкового параметра *.* (то есть любого файла) отключит автозапуск.

Увеличение быстродействия Windows

Увеличение скорости открытия окон

Для того чтобы увеличить скорость открытия окон (всплытия меню), необходимо в разделе реестра HKCU\Control Panel\Desktop создать параметр *MenuShowDelay* типа REG_SZ и присвоить ему значение 0. По умолчанию это значение равно 400. Значение 0 соответствует отсутствию задержки при открытии окон; максимальное значение данного параметра составляет 32 767.

Уменьшение фрагментируемости больших файлов на диске

Для того чтобы операционная система при записи файла на диск сначала нашла для него наиболее подходящее по размеру место и поместила его туда, как можно меньше дробя его на части, в раздел HKLM\System\CurrentControlSet\Control\FileSystem необходимо добавить параметр *ContigFileAllocSize* типа REG_DWORD со значением равным 00000200, который и определяет максимальный размер нефрагментируемого

блока данных на диске. При желании размер такого блока можно увеличить. Данная настройка может быть очень полезной при работе с мультимедиа (уменьшается нагрузка на диск и процессор при записи и воспроизведении видео- или звуковых файлов).

Настройка Boot defrag

Суть функции Boot defrag заключается в дефрагментации тех файлов, которые нужны для старта операционной системы. Выключение этой функции позволит на некоторое время уменьшить время загрузки, но постепенно она будет становиться все медленнее. Если необходимо отключить данную функцию (делать это не рекомендуется!), то в разделе HKLM\SOFTWARE\Microsoft\Dfrg\BootOptimizeFunction нужно отредактировать параметр *Enable*, присвоив ему значение N (по умолчанию функция Boot defrag активирована и параметру присвоено значение Y).

Очистка файла подкачки

Файл подкачки pagefile.sys служит для виртуального увеличения размера используемой оперативной памяти, и в него, по мере необходимости, выгружаются данные из оперативной памяти, а потом подгружаются обратно.

После завершения работы операционной системы в файле подкачки могут оставаться выгруженные в него данные. Для очистки файла подкачки после завершения работы нужно в разделе реестра HKLM\ SYSTEM\ CurrentControlSet\Control\Session Manager\ Memory Management присвоить параметру *ClearPageFileAtShutdown* типа REG_DWORD значение 1.

Настройка режима использования файла подкачки

При наличии в ПК большого объема оперативной памяти можно попытаться повысить производительность системы, запретив выгружать в файл подкачки запущенные системные драйверы и пользовательские коды. Для этого в разделе реестра HKLM\SYSTEM\CurrentControlSet\ Control\Session Manager\Memory Management параметру *DisablePagingExecutive* типа REG_DWORD необходимо присвоить значение 1.

Данную процедуру целесообразно производить только при объеме оперативной памяти более 1 Гбайт.

Ускорение процесса завершения работы Windows

Существует возможность определять величину временн*о*го интервала, в течение которого система должна завершить свою работу. Для этого в разделах реестра HKLM/SYSTEM/CurrentControlSet/Control, HKLM/SYSTEM/CurrentControlSet001/Control и HKLM/SYSTEM/CurrentControlSet002/Control необходимо создать строковый (REG_SZ) параметр *WaitToKillServiceTimeout*, который задает

временную задержку закрытия всех запущенных сервисов в миллисекундах. По умолчанию его значение равно 20 000, то есть 20 с.

Для сокращения времени завершения нужно уменьшить это значение, например, до 10 000, что эквивалентно 10 с. Слишком существенное уменьшение значения этого ключа не позволит системе корректно завершить запущенные сервисы, что может сказаться на стабильности загрузки. В процессе завершения работы система уведомляет все сервисы и дает каждому из них время на корректное закрытие, по истечении которого сервис уничтожается. При этом некоторые параметры настройки сервиса могут быть не сохранены. Следовательно, если при уменьшении этого времени система становится нестабильной, рекомендуется увеличить это значение до достижения системой полной стабильности.

Настройка времени, по истечении которого приложение считается зависшим

Программа считается зависшей, если она не реагирует на обращение к ней. По умолчанию зависшим является приложение, которое не отвечает в течение 5000 миллисекунд. Этот временной интервал можно изменить в разделе реестра HKCU\Control Panel\Desktop, для чего следует отредактировать строковый (REG_SZ) параметр (первоначально его придется создать) *HungAppTimeout*, присвоив ему необходимое значение в миллисекундах (рекомендуемое значение — 1000).

Автоматическое завершение зависших программ

Для того чтобы разрешить системе автоматически завершать зависшие процессы, в разделе реестра HKCU\Control Panel\Desktop (первоначально его придется создать) нужно присвоить строковому параметру *AutoEndTasks* значение 1. Значение 2 соответствует тому, что процессы не завершаются автоматически. Система ожидает, когда процесс завершится, и если время завершения процесса превышает значение параметра HungAppTimeout, то появляется диалоговое окно, указывающее, что приложение зависло.

Настройка времени ожидания перед завершением зависшего приложения

Установить время ожидания перед завершением зависшего приложения можно путем создания в разделе реестра HKCU\Control Panel\Desktop строкового параметра *WaitToKillAppTimeout* и присвоения ему значения времени ожидания в миллисекундах. По умолчанию это время составляет 2000 миллисекунд (рекомендуется — 1000).

Оптимизация работы ядра операционной системы

Для того чтобы предотвратить сбрасывание исполнимого кода ядра операционной системы из оперативной памяти в файл подкачки (что случа-

ется при нехватке оперативной памяти), необходимо в разделе реестра HKLM\System\CurrentControlSet\
Control\SessionManager\MemoryManagement присвоить параметру *DisablePagingExecutive* типа REG_DWORD значение 1.

По умолчанию устанавливается нулевое значение данного параметра, то есть допускается выгрузка исполнимого кода ядра в виртуальную память на жестком диске.

Данную опцию нельзя включать в том случае, если применяется спящий (hibernate) или ждущий (standby) режим компьютера.

Настройка приоритета процессора

Чтобы основные ресурсы процессора были отданы запущенным приложениям, а фоновые задачи имели более низкий приоритет, необходимо отредактировать параметр *Win32PrioritySeparation* типа REG_DWORD в разделе реестра HKLM\ SYSTEM\CurrentControlSet\ Control\PriorityControl.

По умолчанию значение этого параметра равно 2, при том что он может принимать значения в диапазоне от 0 до 26 (hex). При нулевом значении параметра активные и фоновые приложения имеют одинаковый приоритет. При значении 1 активные приложения получают немного больше ресурсов, чем фоновые, при значении 2 — еще больше ресурсов и т.д. (рекомендуемое значение для не очень производительных ПК — 6).

Отключение таблицы совместимости в NTFS

По умолчанию при использовании файловой системы NTFS система создает специальную таблицу совместимости со старыми приложениями, которые могут работать только с файловыми именами в формате MS-DOS (восемь символов имени файла и три символа для его расширения). Данная возможность в настоящее время неактуальна, и ее можно отключить, что повысит производительность системы. Для этого в разделе реестра HKLM\System\CurrentControlSet\ Control\FileSystem параметру *NtfsDisable8dot3NameCreation* типа REG_DWORD нужно присвоить значение 1.

Обновление метки последнего доступа к папке

Если применяется файловая система NTFS, то по умолчанию операционная система обновляет метку последнего доступа к папке при ее открытии. Эта возможность может тормозить систему при слишком большом количестве файлов папок, и, если она вам не нужна, ее можно отключить. Для этого в разделе реестра HKLM\SYSTEM\ CurrentControlSet\ Control\FileSystem установите параметр DWORD *NtfsDisableLastAccessUpdate* типа REG_DWORD равным 1.

Установка способа доступа к общим ресурсам

За настройку способа доступа к общим ресурсам компьютера из сети отвечает параметр *Restrictanonymous* типа REG_DWORD в разделе реестра HKLM\System\CurrentControlSet\ Control\Lsa.

При значении этого параметра, равном 1, запрещен анонимный доступ. Пользователям не разрешено просматривать удаленно учетные записи и общие ресурсы.

При значении 2 запрещен любой неявный доступ к системе (в сетевом окружении компьютер не будет виден, однако доступ к нему можно получить через его IP-адрес).

Сокрытие компьютера в Сети

Для того чтобы компьютер был не виден в сети (не отображался в сетевом окружении), необходимо в разделе реестра HKLM\System\ CurrentControlSet\Services\LanmanServer\Parameters присвоить параметру *Hidden* типа REG_DWORD значение 1.

5.2. Практический блок

Задание 1. **Резервная копия реестра**

1. Создайте копию реестра.
2. Измените обои рабочего стола при помощи реестра.
3. Установите отображение версии Windows на рабочем столе.
4. Восстановите настройки реестра при помощи импорта сохраненного файла. Убедитесь, что данные были восстановлены.

Задание 2. **Настройка компьютера**

1. Создайте точку восстановления системы.
2. При помощи reg-файла выполните следующие действия:

- Добавьте значок Корзины в папку Мой компьютер.
- Добавьте команду Редактор реестра в контекстное меню Мой компьютер. Настройте пункт меню на запуск программы Regedit.
- Отключите отображение стрелок ярлыков.
- Поменяйте раскладку клавиатуры в окне приветствия.

3. Предъявите работу преподавателю.
4. Создайте reg-файл, восстанавливающий все внесенные вами изменеия

Задание 3. **Настройка автоматического входа в систему**

1. Создайте пользователя, назначьте ему логин и пароль.
2. Настройте автоматический вход в систему для нового пользователя.
3. Настройте компьютер так, чтобы при каждом автоматическом входе отображалось сообщение об оставшемся количестве входов (количество входов уменьшается на 1 при каждом входе).

Для отображения при автоматическом входе количества оставшихся входов можно создать .bat-файл, запускающийся автоматически при загрузке системы либо воспользоваться разделом HKLM\SOFTWARE\Microsoft \WindowsNT\CurrentVersion\Winlogon реестра Windows. Для доступа к разделам реестра из командной строки можно воспользоваться утилитой REG:

REG QUERY “раздел реестра” /v “параметр реестра” - выводит имя параметра, его тип и содержащееся в нем значение.

REG ADD “раздел реестра” /v “параметр реестра” /t “тип параметра” /d “значение” - создает параметр с указанным значением в указанном разделе реестра. По умолчанию типа параметра — REG_SZ.

Задание 4. **Настройка безопасности**

1. Создайте общую сетевую папку, убедитесь, что она видна на соседних компьютерах, подключенных к сети.
2. Запретите неявный доступ к системе.
3. Пошлите эхо-запрос на ваш компьютер (используйте команду ping).

Список использованной литературы

1. Анацкая А.Г. Операционные системы и среды: учебно-методическое пособие для студентов специальности 2301013 «Автоматизированные системы управления» всех форм обучения. – Омск: Изд-во Омского экономического института, 2007.

2. Крюков А.А. Расширенное администрирование ОС Unix (Linux). М.: Центр компьютерного обучения "Специалист" при МГУ им. Н. Э. Баумана, 2006.

3. Торчинский Ф.И. Unix. Практическое пособие администратора. - СПб.: Символ-Плюс, 2003.

4. Пахомов С. Справочник по реестру Windows Vista 32-bit // КомпьютерПресс вып. 4, 2008

5. Linux Документация [Электронный ресурс]. URL: http://www.linux.org.ru/books.

6. Основы Slackware Linux [Электронный ресурс] URL: http://www.opennet.ru.

Содержание

Учебное издание

Бречка *Денис Михайлович*

ОПЕРАЦИОННЫЕ СИСТЕМЫ.
ЧАСТЬ 1.
ПАКЕТНЫЕ ФАЙЛЫ И УПРАВЛЕНИЕ КОМПЬЮТЕРОМ
УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ