ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

УПРАВЛЕНИЕ ДИСТАНЦИОННОГО ОБУЧЕНИЯ И ПОВЫШЕНИЯ КВАЛИФИКАЦИИ

Кафедра «Информационные технологии»

Учебно-методическое пособие

по дисциплине

«Операционные системы»

Авторы

Венцов Н.Н.,
Гусев А.А.,
Пушков М.Е.,
Подколзина Л.А.

Ростов-на-Дону, 2017

Аннотация

Методические указания предназначены для студентов направлений подготовки бакалавров 09.03.02 «Информационные системы и технологии» и 09.03.03 «Прикладная информатика».

Авторы

Кандидат технических наук, доцент каф. «ИТ», Венцов Н.Н.;

Senior Java SE7 программист ООО «Открытые бизнес-решения», Гусев А.А.;

Разработчик Oracle DBI ООО «ДиБиАй», Пушков М.Е.;

Программист каф. «ИТ», Подколзина Л.А.

Оглавление

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 1 «УСТАНОВКА LINUX DEBIAN»

Теоретическая часть

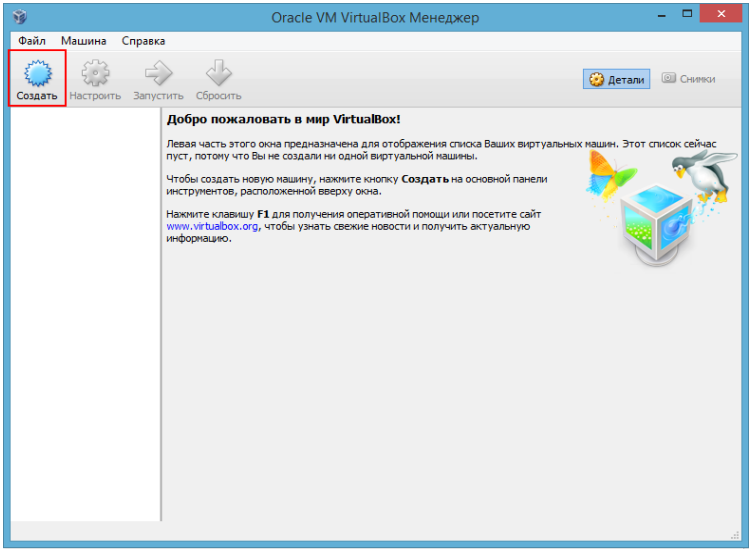
В ходе выполнения этой работы мы установим ОС Linux Debian в среде виртуализации Oracle Virtual Box.

Debian ([ˈdɛbiən]) — это операционная система с открытым исходным кодом, дистрибутив свободного ПО. В настоящее время Debian GNU/Linux — один из самых популярных и важных дистрибутивов GNU/Linux, оказавший значительное влияние на развитие этого типа ОС в целом. Также существуют проекты на основе других ядер: Debian GNU/Hurd, Debian GNU/kFreeBSD и Debian GNU/kNetBSD. ОС Debian может использоваться в качестве операционной системы как для серверов, так и для рабочих станций.

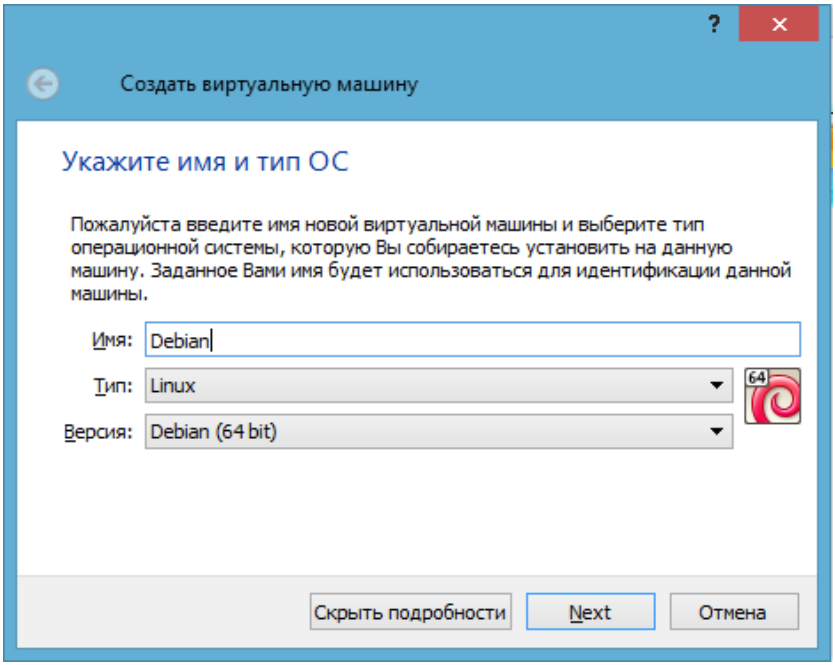
Рекомендации к выполнению работы

Загрузите необходимую версию VB на странице https://www.virtualbox.org/wiki/Downloads и установить ее на ваш ПК.

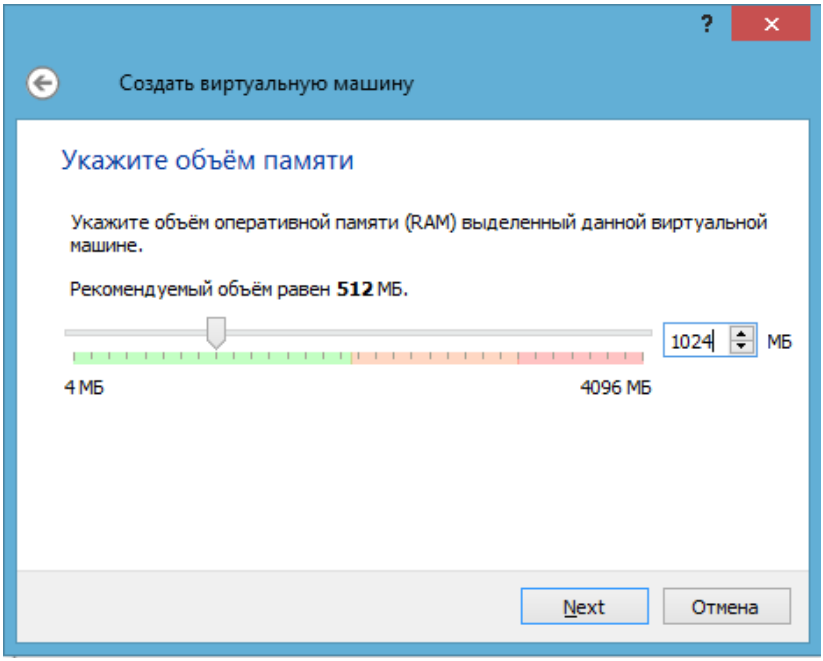
Далее начнем создание виртуальной машины. Нажмите на кнопку «Создать».



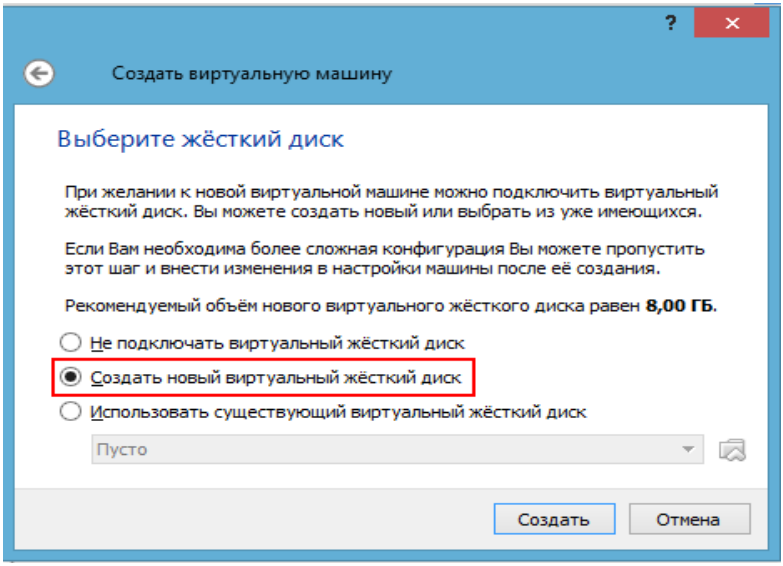
В поле «Имя» введите произвольное имя виртуальной машины (например, Debian). После этого автоматически определится тип операционной системы, но также можно выбрать его вручную, выбрав соответствующие значения полей: «Тип» - Linux, «Версия» - Debian (64 bit) или Debian (32 bit).



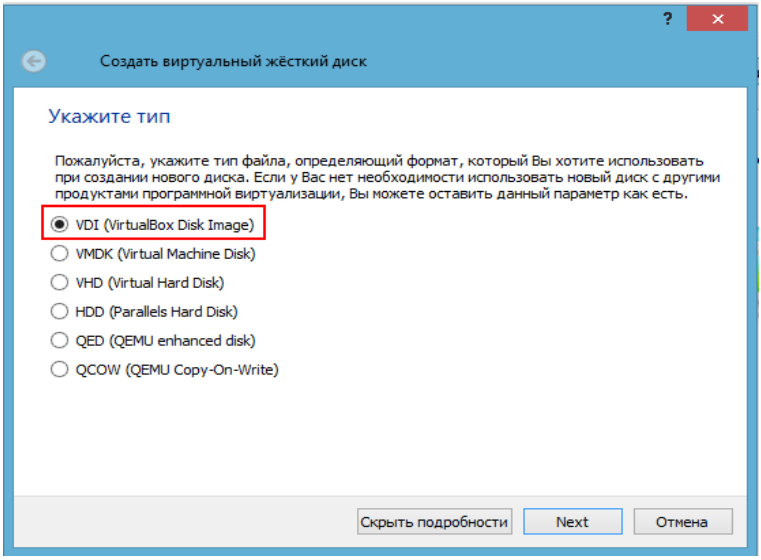
Далее необходимо задать количество оперативной памяти, отведённой под виртуальную машину. Минимальным ее количеством для Debian с графической оболочкой будет 256 mb RAM, для GNOME 3 - 1 gb, поэтому установите это значение.



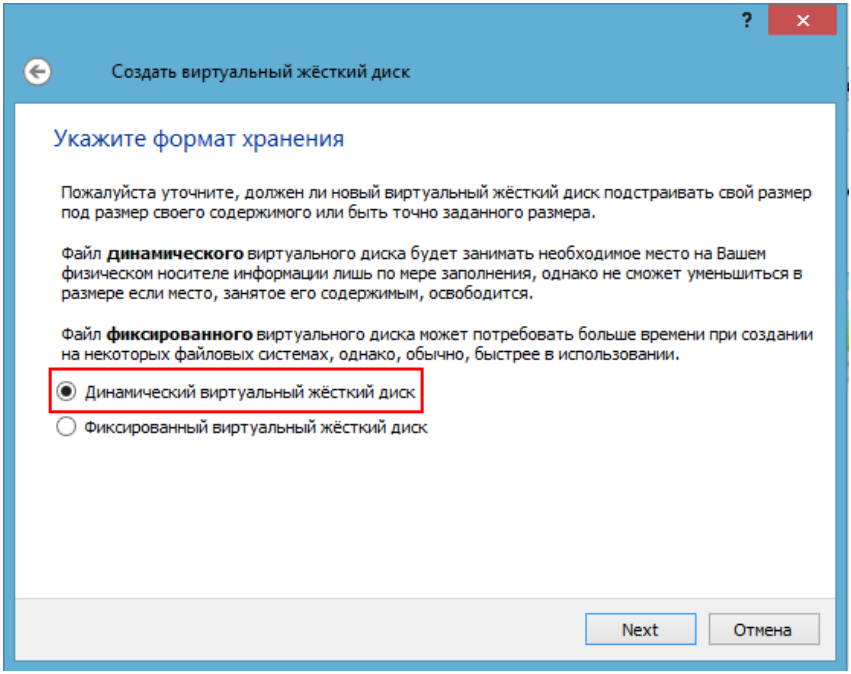
Далее необходимо выбрать виртуальный жёсткий диск, на который будет производиться установка Debian. Выберите опцию «Создать новый виртуальный жёсткий диск»:



Оставьте формат диск выбранным по умолчанию.

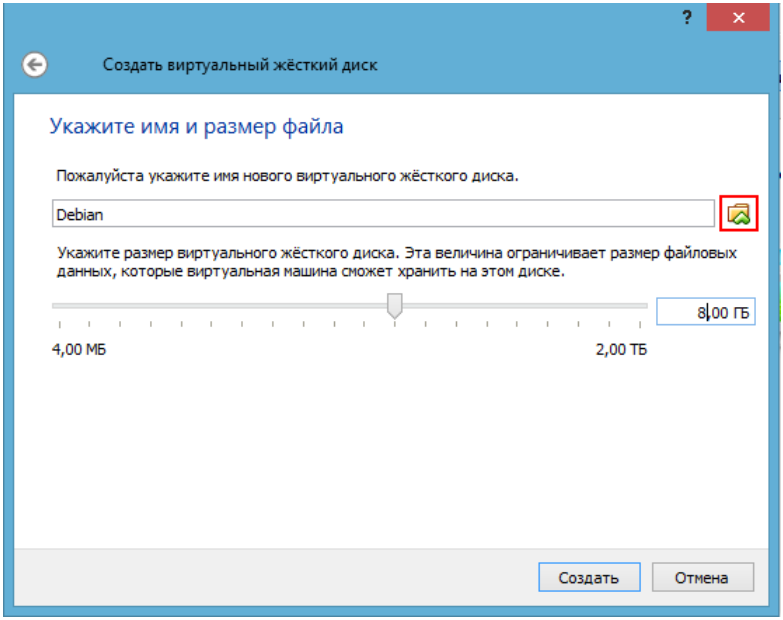


Для выбранного ранее виртуального жесткого диска необходимо указать формат хранения. Подробное описание различий между фиксированным и динамическим виртуальным диском будет показано в окне. Выберите динамический виртуальный диск.



Важно правильно выбрать место хранения виртуального жёсткого диска. В идеальном варианте его следует хранить на отдельном от системного жёстком диске. Это связано с тем, что при выполнении операций чтения или записи на жесткий диск внутри виртуальной машины в момент высокой нагрузки выполнение этих же операций в основной ОС будет значительно замедляться.

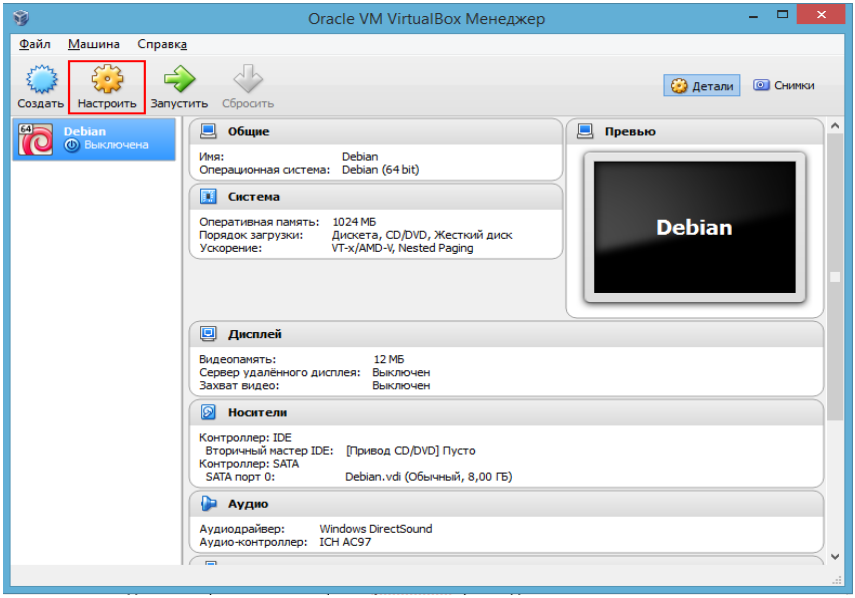
Далее необходимо задать имя нового виртуального жесткого диска и указать его размер. Для учебных целей не понадобится большое количество дискового пространства, поэтому можете оставить значение по умолчанию (8 Gb).



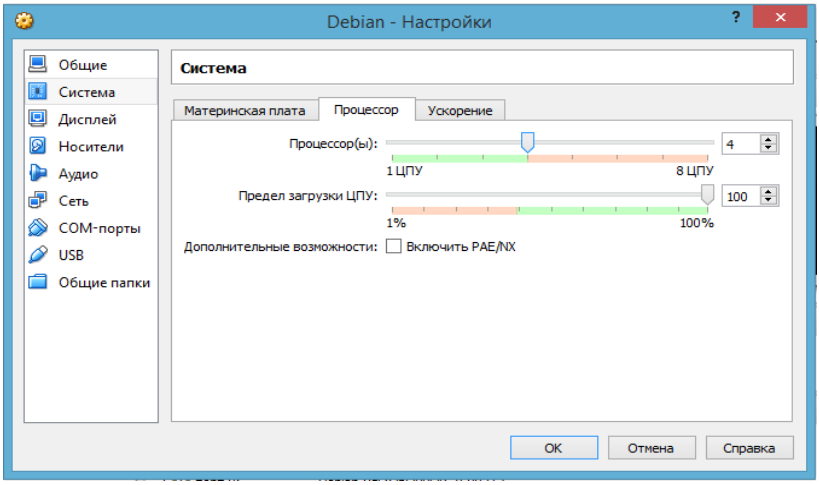
Затем нажмите кнопку «Создать».

После этого виртуальная машина готова к запуску, но необходимо произвести дополнительные настройки для подключения установочного образа и увеличения производительности.

Выберите в списке слева созданную виртуальную машину и нажмите клавишу «Настроить». Откроется окно настроек.

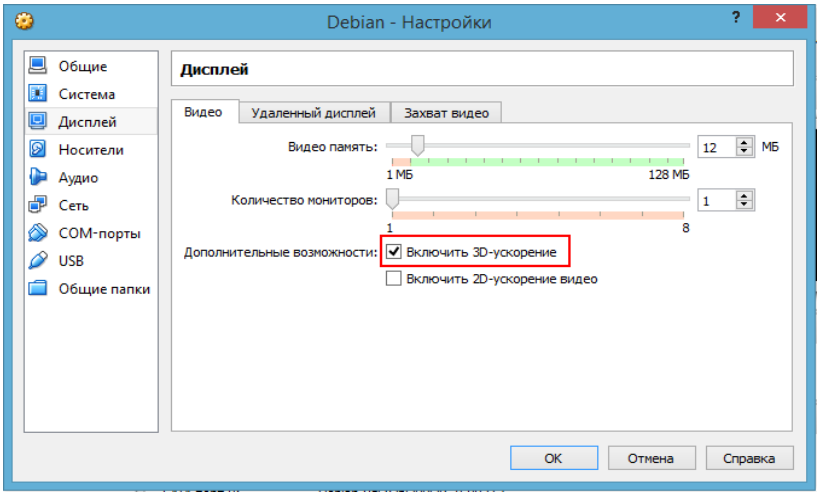


На вкладке Система/Процессор можно задать количество процессоров, которые будет использовать виртуальная машина, предельное значение загрузки ЦПУ.

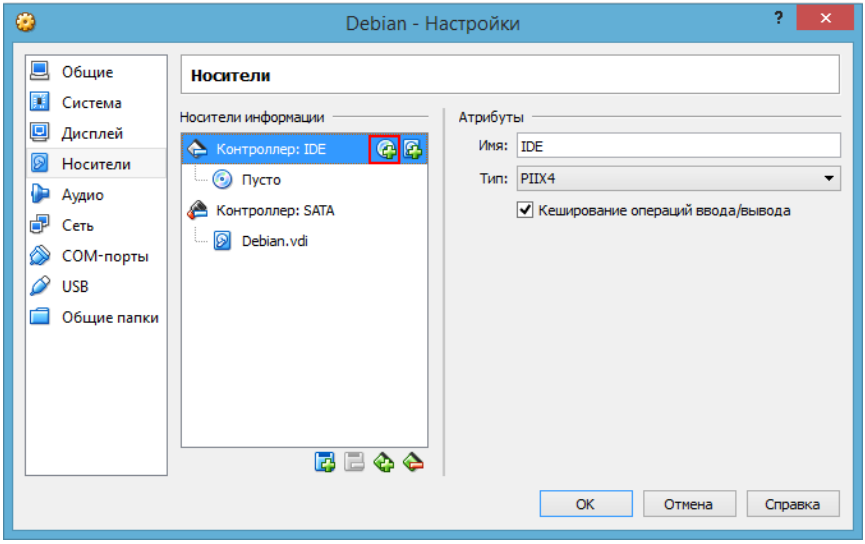


На вкладке Дисплей можно установить объем видеопамяти и включить ускорение 3D графики. Подключение ускорения 2D

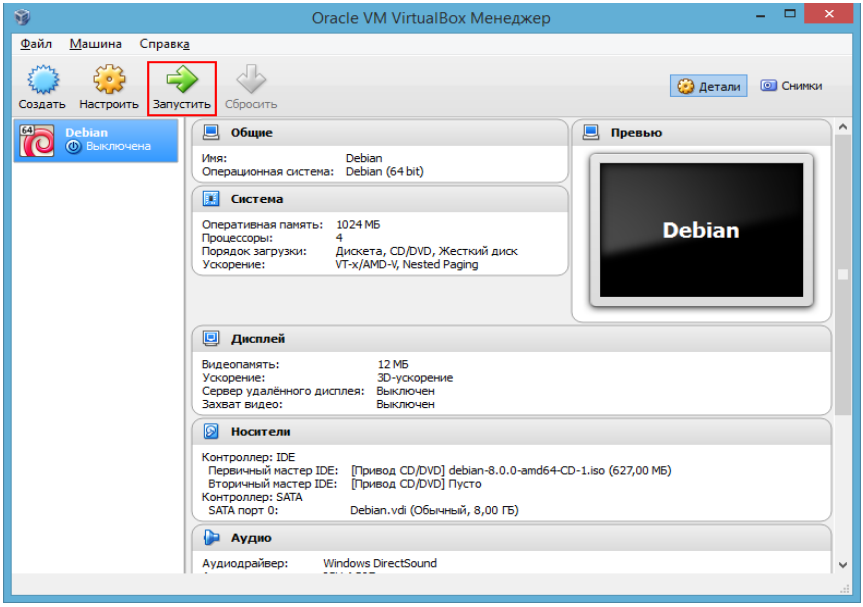
графики не несёт смысла, так как не поддерживается в Linux.



На вкладке Носители, в разделе Контроллер IDE необходимо нажать на кнопку CD диска и выбрать скачанный образ.

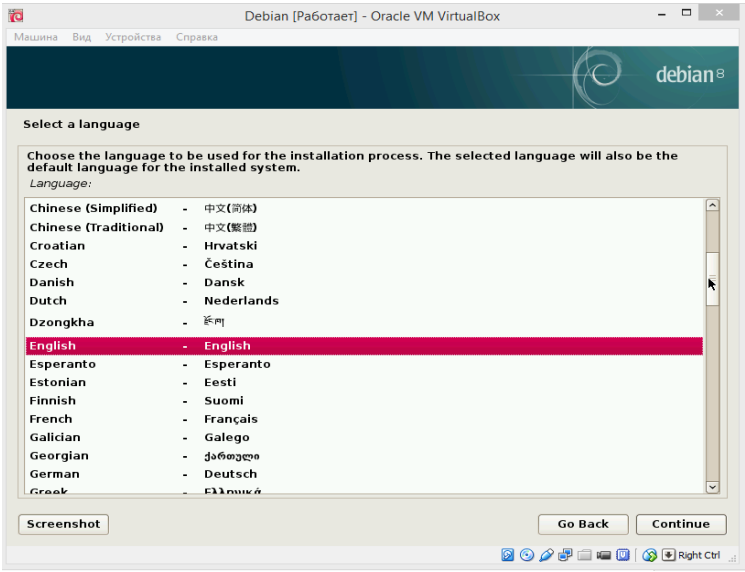


Теперь виртуальная машина готова к запуску. Для этого нажмите кнопку «Запустить».

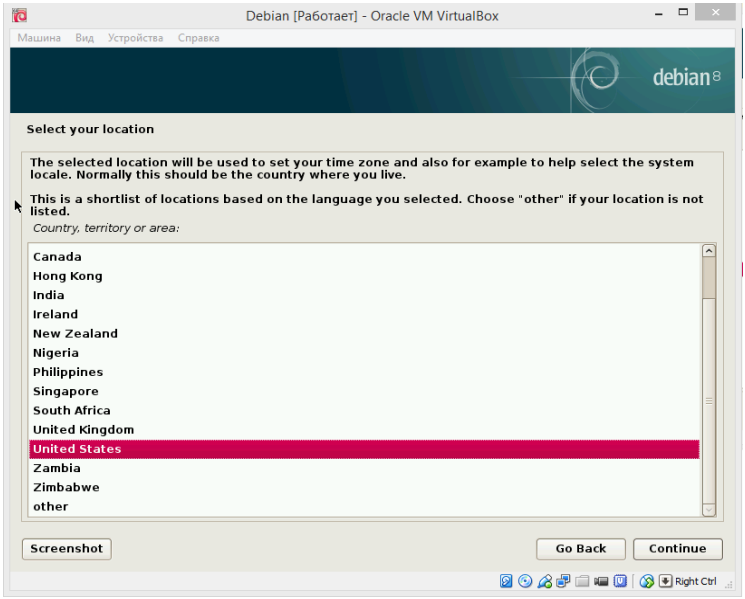


В загрузочном окне Debian выберите Graphical install.

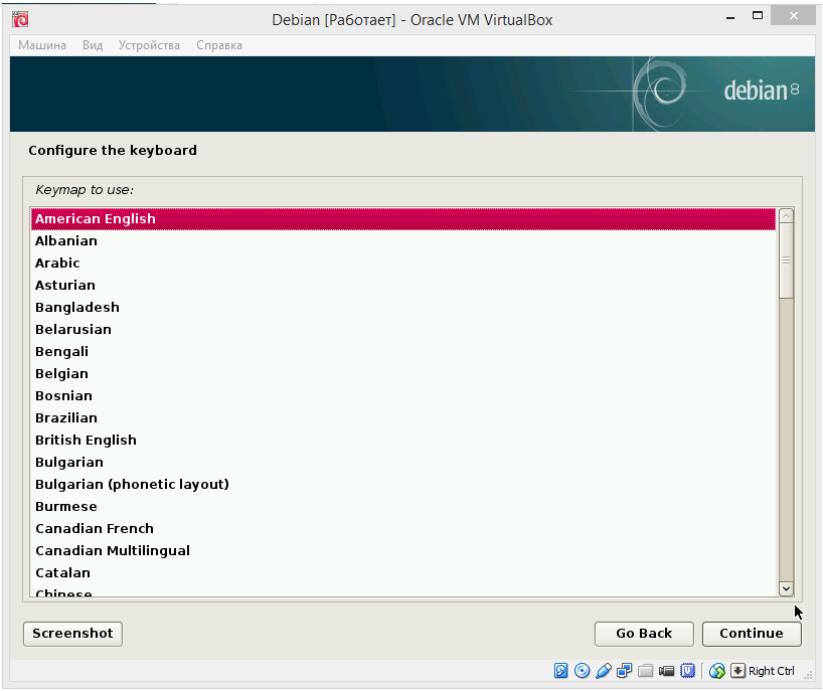
В появившемся окне выберите желаемый язык ОС.



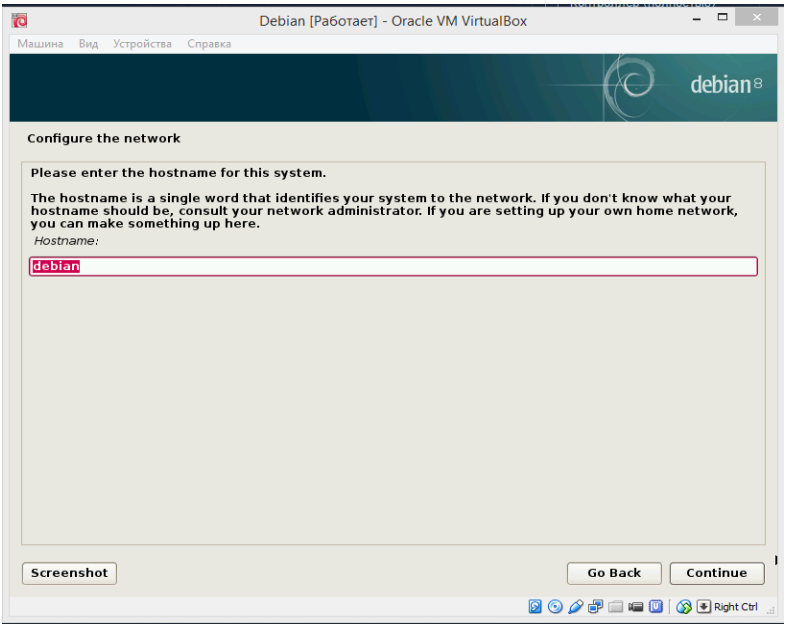
Также укажите свое местоположение.



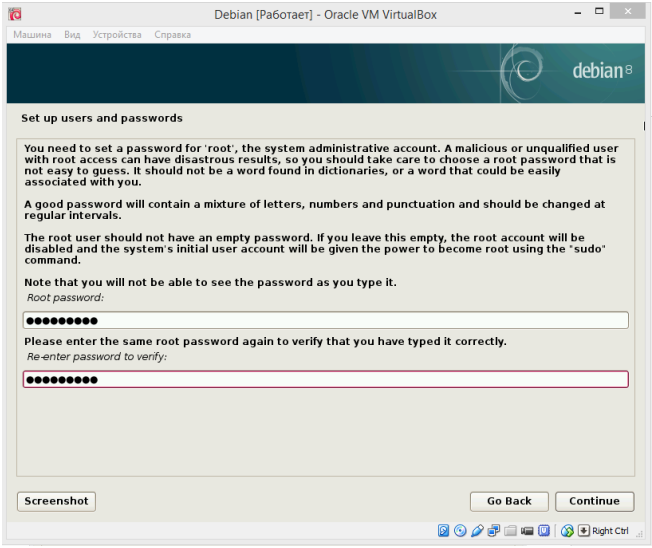
Выберите раскладку клавиатуры.



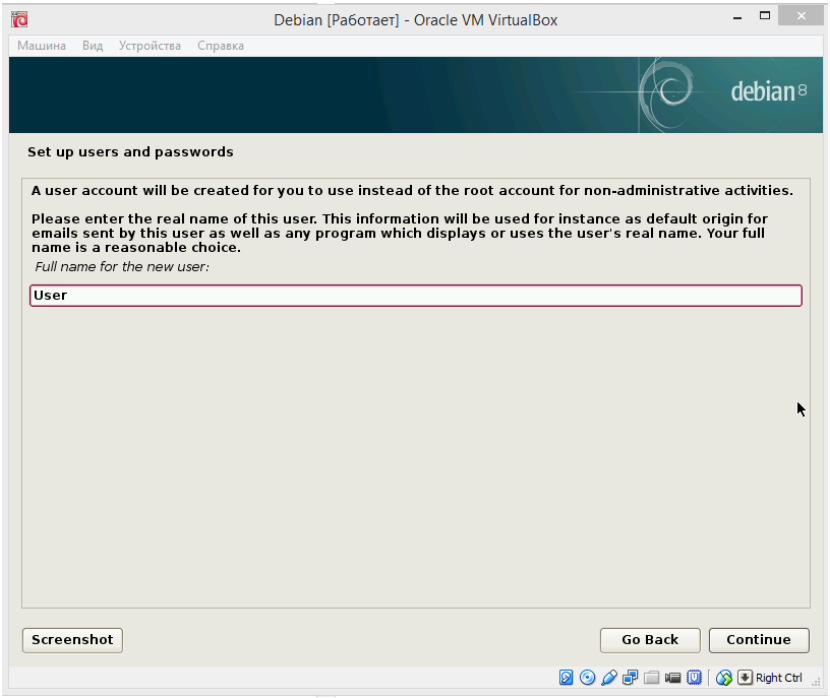
Укажите имя хоста.



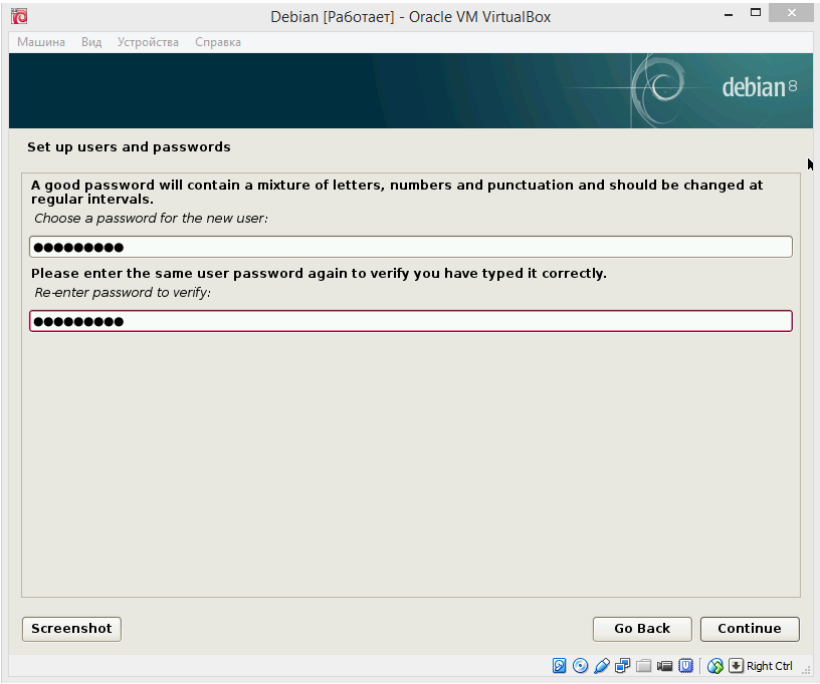
Теперь задайте пароль для Root-пользователя.



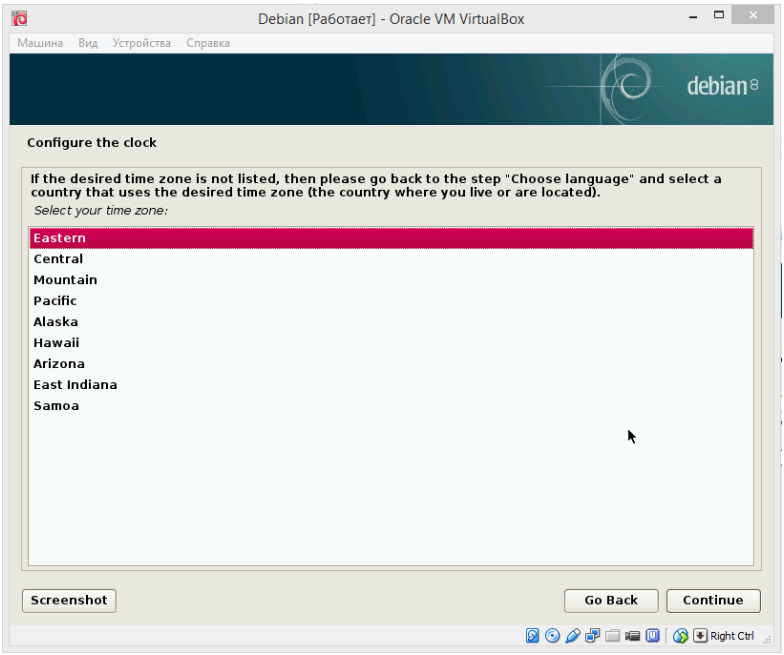
Далее необходиом создать новую учетную запись для работы в ОС. Введите в появившемся окне имя нового пользователя.



Затем задайте для созданного пользователя пароль.

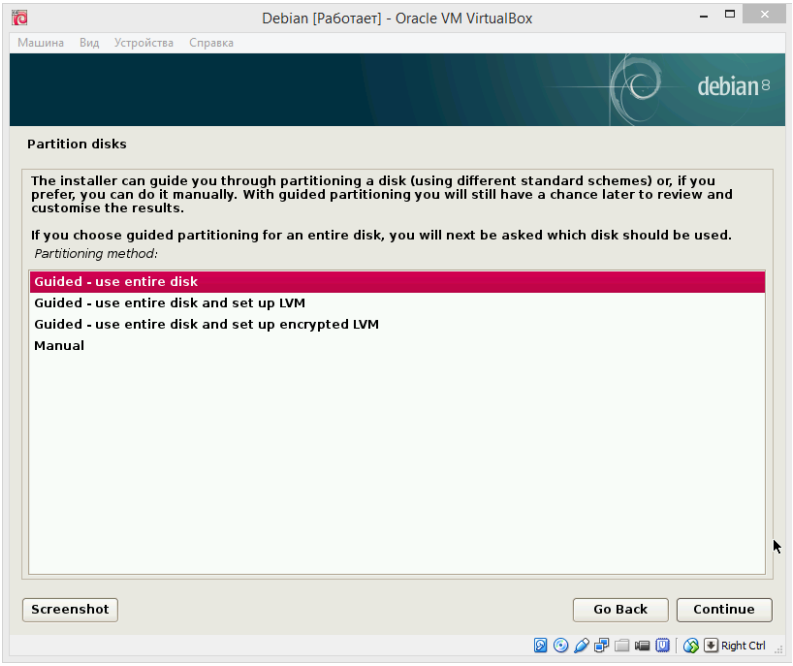


И определите вашую временную зону (часовой пояс).

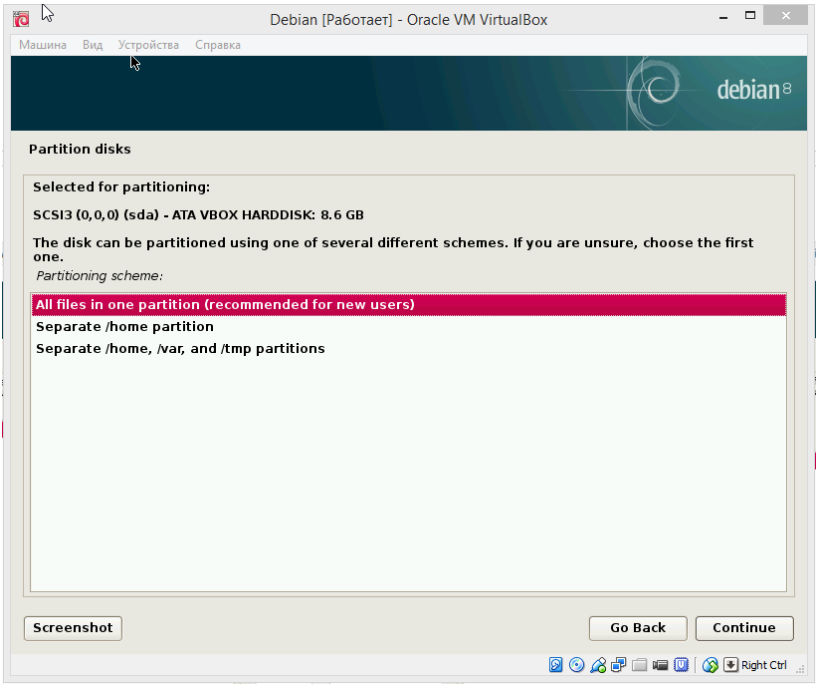


Следующим этапом станет разметка диска.

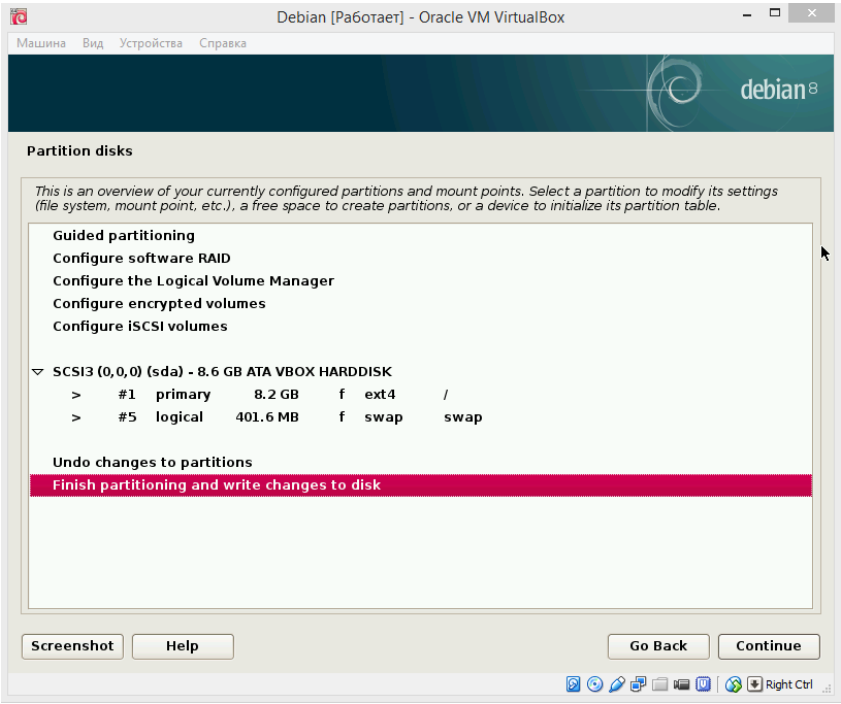
Так как на жёстком диске будет установленна только одна ОС, выберите первый метод. В этом случае ОС Debian будет использовать целиком весь диск.



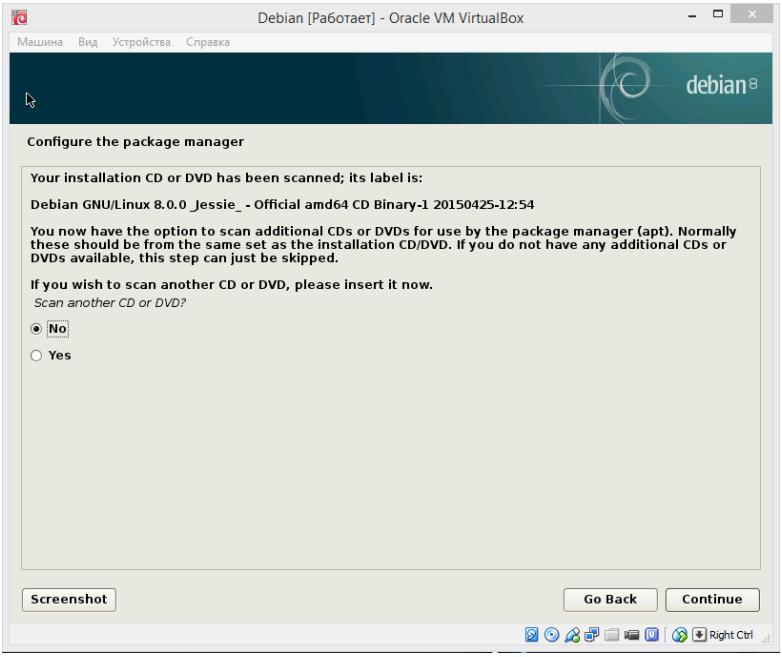
В качестве схемы разметки выберите первую, при которой все файлы будут храниться в одном разделе.



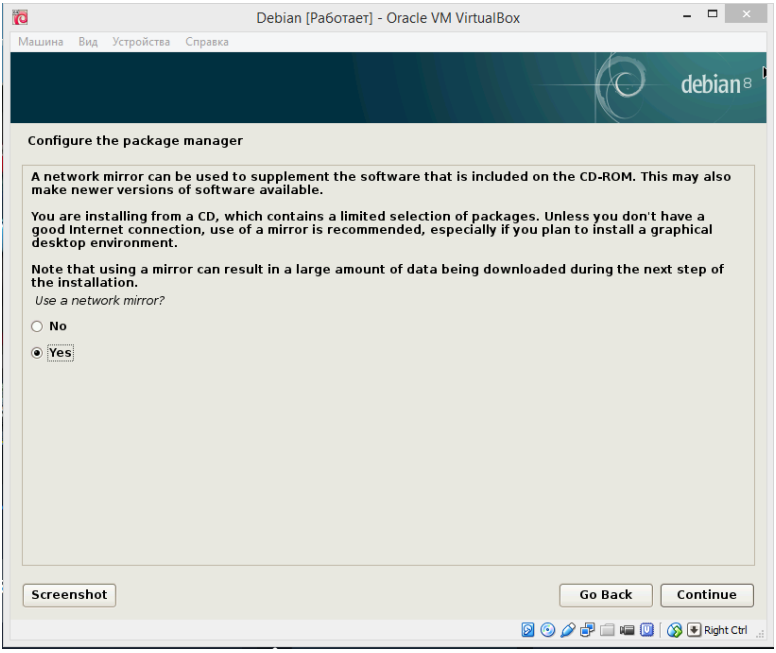
После этого появится окно, в котором будет наглядна продемонстрирована будущая разметка.

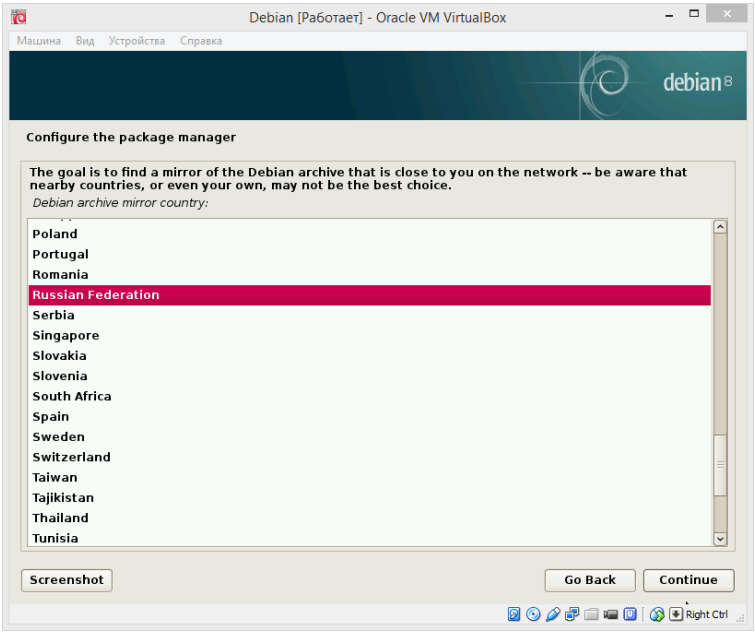


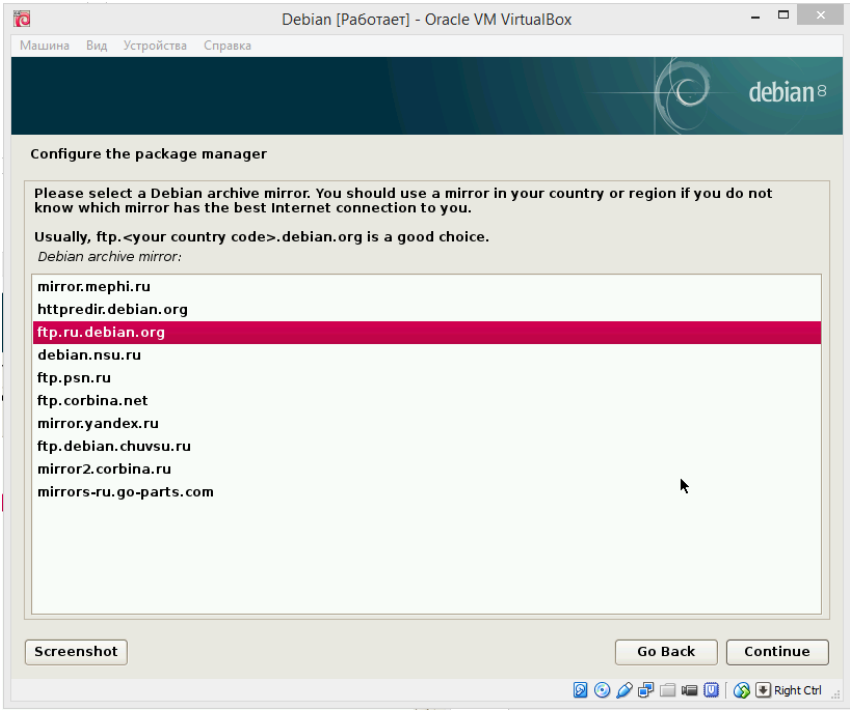
Если имеются диски, содержащие другие пакеты, установите их, выбрав вариант «Yes». В противном случае выберите «No».



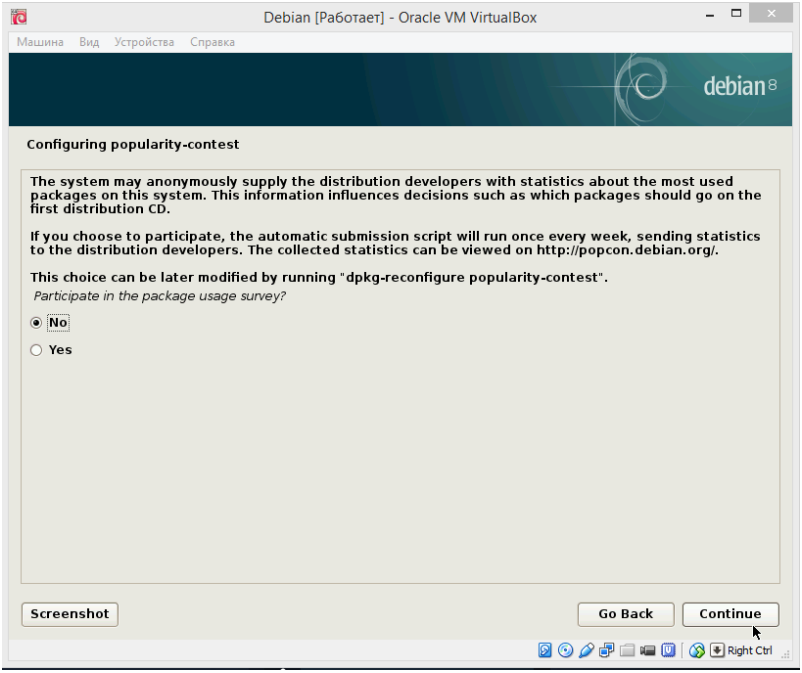
На следующем этапе следует выполнить настройку зеркала для установки необходимых пакетов.


Debian [Работает] - Oracle VM VirtualBox
Машина Вид Устройства Справка
debian 8
Configure the package manager
A network mirror can be used to supplement the software that is included on the CD-ROM. This may also make newer versions of software available.
You are installing from a CD, which contains a limited selection of packages. Unless you don't have a good Internet connection, use of a mirror is recommended, especially if you plan to install a graphical desktop environment.
Note that using a mirror can result in a large amount of data being downloaded during the next step of the installation.
Use a network mirror?
No
Yes
Screenshot
Go Back
Continue
Right Ctrl

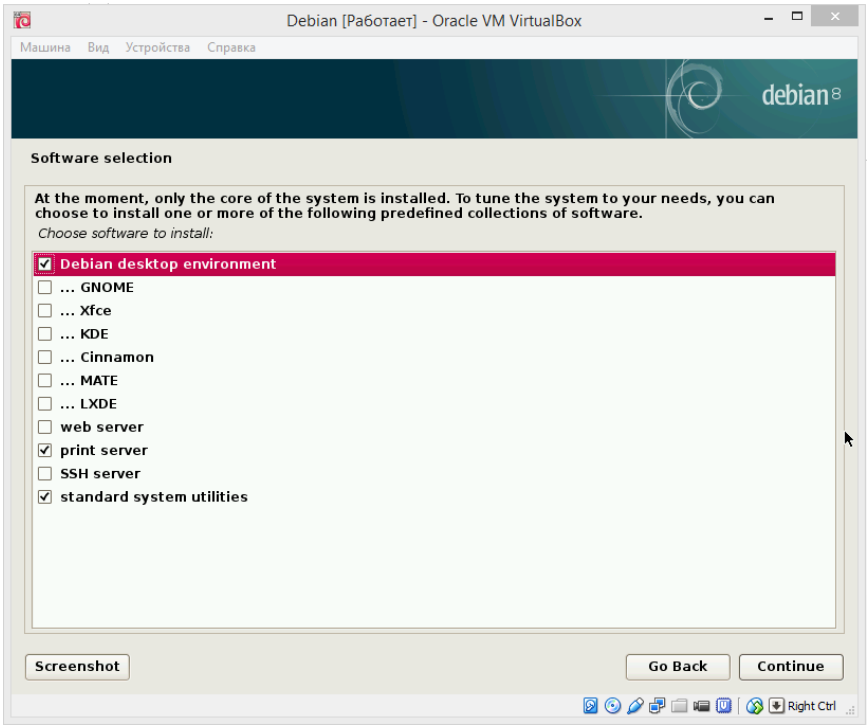

Debian [Работает] - Oracle VM VirtualBox
Машина Вид Устройства Справка
debian 8
Configure the package manager
The goal is to find a mirror of the Debian archive that is close to you on the network -- be aware that nearby countries, or even your own, may not be the best choice.
Debian archive mirror country:
Poland
Portugal
Romania
Russian Federation
Serbia
Singapore
Slovakia
Slovenia
South Africa
Spain
Sweden
Switzerland
Taiwan
Tajikistan
Thailand
Tunisia
Screenshot
Go Back
Continue
Right Ctrl


Debian [Работает] - Oracle VM VirtualBox
Машина Вид Устройства Справка
debian 8
Configure the package manager
Please select a Debian archive mirror. You should use a mirror in your country or region if you do not know which mirror has the best Internet connection to you.
Usually, ftp.<your country code>.debian.org is a good choice.
Debian archive mirror:
mirror.mephi.ru
httpredir.debian.org
ftp.ru.debian.org
debian.nsu.ru
ftp.psn.ru
ftp.corbina.net
mirror.yandex.ru
ftp.debian.chuvsu.ru
mirror2.corbina.ru
mirrors-ru.go-parts.com
Screenshot
Go Back
Continue
Right Ctrl

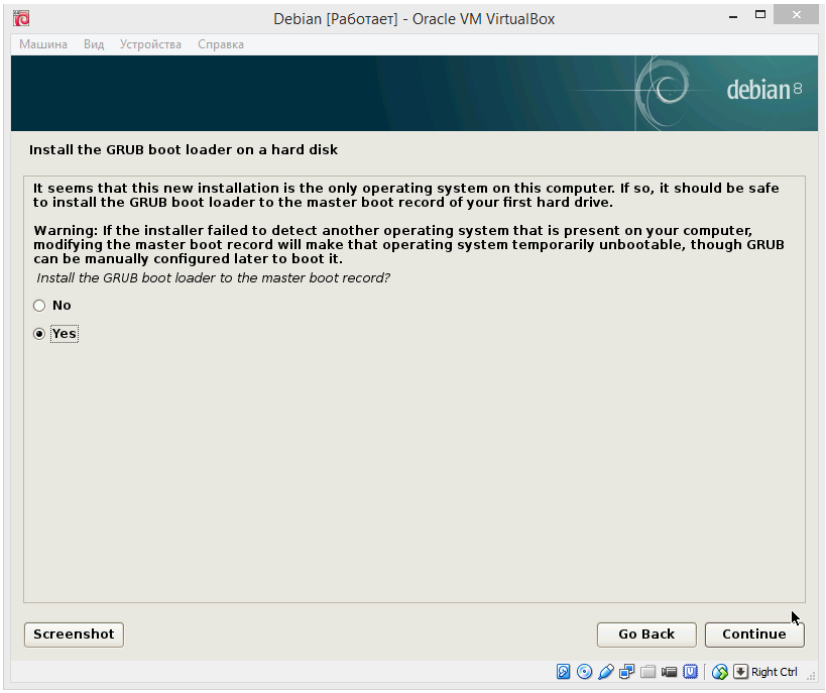
Далее можно выбрать, будет ли система отправлять анонимную статистику.



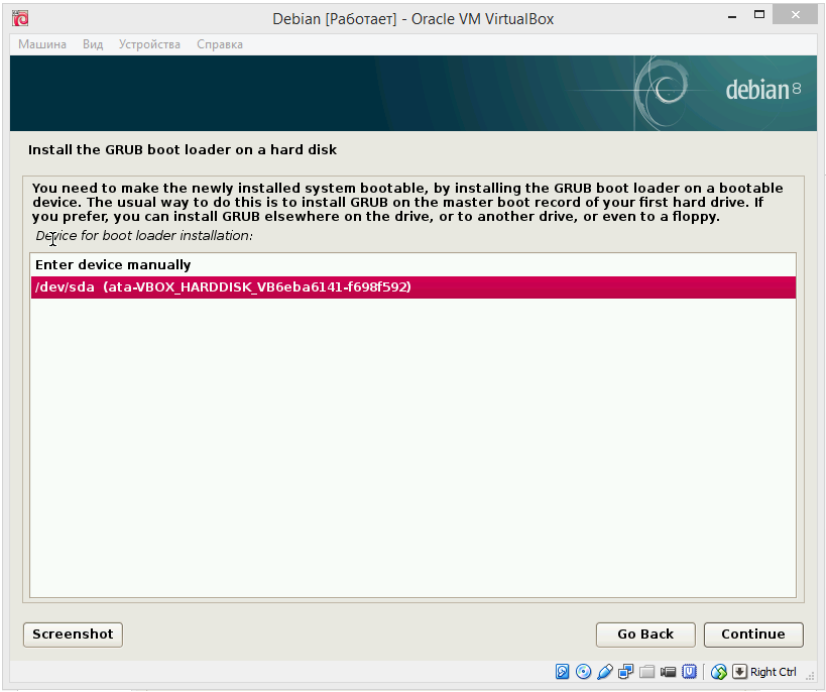
Также необходимо выбрать оконное окружение и базовый набор программ.



Далее система предложит установить загрузчик GRUB в MBR (Mastee Boot Record) жёсткого диска. Согласитесь с этим.



Затем выберите жёсткий диск для установки GRUB.



На этом установка окончена.

Контрольные вопросы

1. Установку какой ОС мы проводили на этом занятии и в какой среде виртуализации?
2. Что такое Debian?
3. Что такое файл динамического виртуального диска?
4. Что такое файл фиксированного виртуального диска?

Список использованных источников

1. Debian // Википедия - свободная энциклопедия URL: https://ru.wikipedia.org/wiki/Debian (дата обращения: 09.09.2015).

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 2. «ЗНАКОМСТВО С ОС RED HAT LINUX»

Теоретическая часть

Red Hat Enterprise Linux – это серверная операционная система на аппаратной платформе Intel x86, EM64T, Itanium2 и AMD64. Она поддерживает одно- и двухпроцессорные системы с оперативной памятью до 16Gb и может использоваться для самых разнообразных задач: от базовых сетевых сервисов до ведомственных серверов среднего уровня.

В Red Hat Enterprise Linux входят серверные приложения для организации:

- web-серверов;
- сетевых сервисов (DHCP, DNS, Firewall и пр.);
- почтовых серверов;
- серверов печати;
- файловых серверов;
- небольших и средних баз данных.

Все эти компоненты уже включены в дистрибутив, что избавляет от необходимости покупать их отдельно. Это делает Red Hat Enterprise Linux значительно более дешевым решением, чем проприетарные аналоги. Кроме того, при использовании Red Hat Enterprise Linux отсутствует такое понятие, как клиентские лицензии (Device/User CAL), а значит количество работающих пользователей не ограничено.

Загрузка системы

При включении питания ПК BIOS проводит тестирование оборудования, ищет загрузочное устройство, передает управление найденному загрузчику, который, в свою очередь, начинает загрузку операционной системы.

ОС Linux Red Hat может загружаться с самых разных устройств, начиная от жестких дисков и дискет, заканчивая USB-накопителями и загрузкой по сети. Стандартным для Linux Red Hat загрузчиком является LILO (LInux LOader), однако, как правило, каждый дистрибутив этой ОС предлагает ещё один-два альтернативных варианта.

После выбора операционной системы загрузчик передает управление ядру ОС. Ядро (kernel) начинает проверку оборудования, отображая на экране информацию о ходе проверки. После окончания проверки оборудования ядро Linux Red Hat может быть

переведено в режим интерактивной загрузки. Если пользователь не включил данный режим, то система продолжит автоматическую загрузку сервисных программ, считывая информацию из конфигурационных файлов (/etc/inittab, /etc/rc.d). Процесс и результаты загрузки сервисов так же отобразятся на экране. Если система настроена на загрузку в графическом режиме, то в числе сервисов будет загружен X-сервер, иначе - система будет загружена в текстовом режиме и на экране появится приглашение для авторизации пользователя.

Это означает, что загрузка ядра Red Hat Linux завершена и система ждет ввода имени пользователя для начала сеанса работы.

Работа в командной строке

В только что установленной системе имеется всего один пользователь, который может выполнить вход. Это привелегированный пользователь (суперпользователь) root, наделенный правами администратора.

ВАЖНО! Для пользователя с root-правами нет ограничений по управлению ресурсами системы. Именно поэтому с точки зрения безопасности не следует выполнять повседневные задачи в сеансе суперпользователя.

После ввода имени пользователя, система запросит пароль для него и, если он введен корректно, откроет пользовательский сеанс. Экран примет примерно такой вид:

```
Red Hat Linux release 6.2
Kernel 2.2.14-xx on an i686
localhost login: root
Password:
Last login: Thu Sep 8 11:59:42 on tty3
[root@localhost ~]#
```

С этого момента система готова принимать команды от пользователя и выполнять их. Все команды поступают на исполнение через командную строку (строку приглашения). Строка приглашения – это пользовательский интерфейс, представляемый оболочкой системы. Оболочка – это программа-посредник между пользователем и операционной системой. Оболочки упрощают работу пользователя, представляя такие возможности, как автоподстановка текста, история ввода, встроенные скриптовые языки. Основная их задача – получить команду, введенную пользователем и передать ее на исполнение операционной системе. Формат ввода команд прост и одинаков для всех оболочек: нужно

указать имя команды и при необходимости набор параметров для нее. Пример ввода и выполнения команды без параметров:

[root@localhost ~]# pwd
/root
[root@localhost ~]#

Пример команды с указанием именем файла в качестве параметра:

[root@localhost ~]# cat hello.txtHello,World

Следующий, чуть более сложный пример, иллюстрирует работу с командной строкой Linux Red Hat для компиляции Java-приложений:

[root@localhost ~] javac -classpath /usr/share/tomcat/lib/tomcat-servlet-3.0-api.jar:classes /srv/tomcat/webapps/myapp/WEB-INF/classes/MyServlet.java

Linux Red Hat – это POSIX-совместимая операционная система, то есть она соответствует стандартам и спецификациям IEEE 1003.x (POSIX). Таким образом, набор основных команд Linux Red Hat соответствует командам UNIX-подобных ОС. Подробную информацию о командах вы можете получить через встроенную справочную систему формата man (от manuals) или info. Для получения справки достаточно ввести man (или info) с именем нужной команды в качестве параметра:

[root@localhost ~]# man pwd

Результатом выполнения станет вывод информации о назначении, синтаксисе и ключевых параметрах заданной команды:

PWD(1) User Commands PWD(1)
NAME
pwd - print name of current/working directory
SYNOPSIS
pwd [OPTION]
DESCRIPTION
NOTE: your shell may have its own version of pwd which will supercede the version

described here. Please refer to your shell’s documentation for details about the options it <...>

Аналогичным образом можно получить и «справку о справке»:

[root@localhost ~]# man man

При использовании дополнительных опций, команда man может выполнять дополнительные функции при отображении справочной информации.

stilo:/home/aag # man --help
использование: man [-c|-f|-k|-w|-tZT устройство] [-i|-I]
[-adlhu7V] [-Мпуть] [-Рпейджер]
[-Сфайл] [-Ссписок] [-mсистема] [-pстрока] [-Lлокаль] [-eрасширение]
[раздел] страница ...
-a, --all - найти все подходящие страницы руководств.
-d, --debug - показывать отладочные сообщения.
-e, --extension - ограничить поиск файлами с выбранным расширением.
-w, --where, --location - показывать физическое расположение man страниц.
--location-cat - показывать физическое расположение cat file(s).
-l, --local-file - рассматривать аргумент <страница> как локальные имена файлов.
-u, --update - включить проверку целостности кэша.
-i, --ignore-case - регистронезависимый поиск страниц (включен по умолчанию).
-I, --match-case - регистрозависимый поиск страниц.
-c, --catman - реформатирование устаревших cat страниц.
-7, --ascii - показывать ASCII символы.
-E, --encoding - использовать заданное nroff устройство.
-t, --troff - использовать groff для форматирования страниц.
-T, --troff-device <устройство> - использовать groff с заданным устройством.
-H, --html - использовать lynx для показа html.
-Z, --ditroff - использовать groff и заставить его создавать ditroff.
-X, --gxditview - использовать groff и показать с помощью gditview (X11):
-X = -TX75, -X100 = -TX100, -X100-12 = -TX100-12.
-D, --default - сбросить все параметры в значения по умолчанию.
-C, --config-file <файл> - использовать файл настроек пользователя.
-M, --manpath <путь> - установить путь поиска страниц руководства.
-P, --pager <пейджер> - использовать заданный пейджер для просмотра.
-S, --sections <список> - использовать список разделов, отделённых двоеточием.

-m, --systems <система> - поиск man страниц в других Unix системах.

-L, --locale <локаль> - поиск man страниц только для определённой локали.

-p, --preprocessor - указать препроцессы для запуска.

-V, --version - показать версию.

-h, --help - показать это сообщение.

Выход из справочной системы в командную строку выполняется клавишей q[uite].

Помимо информации о командах, справку можно получить и о системных сервисах (в терминологии Unix – демонах, daemon's), а также о формате служебных и конфигурационных файлов.

Однако не вся информация справочной системы man является локализованной, то есть переведенной на русский язык. В таком случае можно воспользоваться справкой формата info, но она содержит сведения не обо всех командах.

Еще одна возможность получения информации о команде - это выполнение ее с параметром --help. Например:

[root@localhost ~]# pwd --help

Использование: ls [КЛЮЧ]... [ФАЙЛ]...

Выдает информацию о FILE (текущий каталог по умолчанию).

Сортирует в алфавитном порядке, если ни один из ключей -cftuSUX --sort не задан.<...>

В квадратных скобках ([]) указываются НЕОБЯЗАТЕЛЬНЫЕ параметры, а ключи могут быть объединены.

Завершение работы

Первые Unix-подобные ОС создавались в расчете на длительную работу без выключения компьютера. Поэтому завершение работы компьютера с ОС Linux Red Hat имеет некоторые особенности, которые зависят от конфигурации системы. Как правило, полное выключение может выполнять только суперпользователь по команде shutdown. Остальные могут выполнить только завершение собственного сеанса (командой exit или комбинацией клавиш Ctrl+D).

По команде shutdown в зависимости от параметров система может быть остановлена в указанное время с предварительной рассылкой сообщения. Перед выключением ОС записывает все несохраненные данные на диск, выгружает запущенные программы и последовательно останавливает системные сервисы. Во из-

бежание потери информации необходимо дождаться сообщения о том, что питание можно отключить.

При завершении сеанса ОС выгружает только те программы, которые были запущены текущим пользователем, затем выгружает оболочку и снова выводит приглашение login.

Задания для самостоятельной работы

1. При включении компьютера определить, какой загрузчик используется в системе, и установлены ли на нем другие операционные системы.
2. Наблюдая за процессом загрузки ядра, обратить внимание на то, какие сервисы загружаются и сделать предположения о их назначении. Выяснить, как можно перейти в режим интерактивной загрузки. Переключиться в этот режим и определить, какие сервисы могут быть запущены по запросу пользователя.
3. Войти в систему с учетной записью суперпользователя.
4. Ознакомиться со справочными системами man и info.
5. Получить справочную информацию о командах *useradd, passwd, exit, logout, who, shutdown, su, users, groups*.
6. Создать собственную учетную запись и установить для нее пароль.
7. Завершить сеанс суперпользователя.
8. Войти в систему под собственной учетной записью.
9. С помощью справочной системы man проверить предположения о назначении служб, которые запускаются в процессе загрузки системы (см. п.2).

Контрольные вопросы

1. Что такое Red Hat Enterprise Linux?
2. Какие серверные приложения для организации входят в Red Hat Enterprise Linux?
3. С каких устройств может загружаться ОС Linux Red Hat?
4. Что такое root?
5. Что такое POSIX-совместимая операционная система?
6. Что делает команда shutdown?

Список использованных источников

1. Войтов Н.М. Администрирование Red Hat Enterprise Linux. - М.: ДМК Пресс, 2011. - 192 с.

2. Собелл М.Г. Практическое руководство по Red Hat Linux: Fedora Core и Red Hat Enterprise Linux. - 2 изд. - М.: Вильямс, 2005. - 1072 с.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 3. «ТЕРМИНАЛЫ И ТЕКСТОВЫЙ РЕЖИМ»

Теоретическая часть

Текстовый режим Red Hat Linux, несмотря на внешнее сходство с ОС DOS, принципиально отличается от нее по своим возможностям. Дело в том, что даже в текстовом режиме Red Hat Linux, как и все unix-подобные системы, остается многозадачной системой. Более того, изначально созданная как многопользовательская система, Red Hat Linux позволяет одновременно работать различным пользователям. Для обеспечения такой возможности используется концепция *терминалов*.

Терминал - это интерфейс ввода/вывода, состоящий из физических устройства ввода (клавиатура) и вывода (дисплей). Терминал предназначен исключительно для ввода информации и ее отображения на экране. Терминалы бывают физическими (реальными), виртуальными и псевдотерминалами (то есть программами, которые «притворяются» терминалами). При выполнении лабораторных работ будут использоваться виртуальные терминалы.

Red Hat Linux представляет доступ к шести текстовым терминалам, которые соответственно называются tty1, tty2 и т.д. Переключение между ними осуществляется сочетанием клавиш Ctrl+F1, Ctrl+F2, Ctrl+F3 и т.д. При загруженной графической оболочке открытие терминалов и переключение между ними производится клавишами Alt+Ctrl+F<n>, (где n - номер терминала). Сама графическая оболочка будет доступна по Alt+Ctrl+F7. Для работы в каждом новом терминале необходимо авторизоваться. Таким образом, в системе одновременно могут работать несколько различных пользователей, каждый в своем терминальном сеансе.

Первый виртуальный терминал системы вы можете видеть сразу после загрузки ОС. Именно в нем отображается приглашение для авторизации.

Для того чтобы узнать номер текущего терминала, можно использовать команду tty (см. man tty). Команда очень проста и не требует параметров. Пример работы tty:

```
[aag@localhost fpk]$ tty
/dev/tty1
```

С помощью команды who можно определить, в каких терминалах, какие пользователи и в какое время были авторизованы в системе:

```
[aag@localhost fpk]$ who
root tty1  Feb 17 17:11
aag tty2  Feb 17 17:12
aag tty5  Feb 17 17:40
```

Иногда возникает необходимость выполнить некоторые действия от имени другого пользователя (например, от имени root). Для этого не обязательно открывать новый терминальный сеанс, существует команда su. Эта команда открывает сеанс введенного через параметр пользователя, выполняющийся внутри сеанса непривилегированного пользователя. При отсутствии параметра запускается сеанс суперпользователя. Пример выполнения команды su:

```
[aag@localhost ~]$ su
Password:
[root@localhost aag]#
[aag@localhost ~]$ su stud
Password:
[stud@localhost ~]$
```

Для окончания пользовательского сеанса, запущенного командой su, или выхода из терминала, используется команда exit или комбинация клавиш Ctrl+D. В любом случае система закроет сеанс и отобразит строку приглашения.

Дополнительная информация об указанных командах доступна в справочном руководстве формата man или info.

Задания для самостоятельной работы

1. Загрузить систему в текстовом режиме и войти с собственной учетной записью.
2. Выяснить, какой каталог является текущим (см. man pwd).
3. Выяснить, в каком терминале выполняется текущий сеанс.
4. Открыть новый сеанс в tty3 для пользователя root.
5. Повторить п.2
6. Открыть новый сеанс в tty5 для пользователя stud (если такого пользователя нет в системе, то его нужно добавить).
7. Повторить п. 2
8. Перейти в tty1 и выяснить, какие пользователи в каких терминалах работают в текущий момент.
9. Создать текстовые файлы с именем user.txt произвольного содержания в каталогах /root , $HOME , /home/stud.

Для создания файлов Вы можете использовать, к примеру, встроенный редактор файлового менеджера MC (Midnight Commander).

10. Перейти в tty3 и повторить п.9 (имя для файлов – root.txt).
11. Перейти в tty 5 и повторить п.9 (имя для файлов – stud.txt).
12. Сравнить результаты выполнения пп 9, 10 и 11.
13. Перейти в tty1 и объединить содержимое файлов, полученных в ходе выполнения пп 9, 10 и 11 в один файл с именем all.txt и вывести этот файл на экран (см. man cat).
14. Повторить предыдущее задание от имени суперпользователя из tty1.
15. Завершить все сеансы.

Контрольные вопросы

1. Что такое терминал?
2. К скольким терминалам по умолчанию предоставляет доступ Red Hat Linux и как они называются?
3. Какую команду нужно использовать для того, чтобы узнать номер текущего терминала?

Список использованных источников

1. Баркакати Н. Red Hat Linux. Секреты профессионала. - М.: Вильямс, 2004. - 1056 с.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 4. «ИЗУЧЕНИЕ СТРУКТУРЫ ФАЙЛОВОЙ СИСТЕМЫ ОС LINUX»

Теоретическая часть

В ходе выполнения данной работы будут изучены структуры файловой системы ОС LINUX, команды создания, удаления, модификации файлов и каталогов, функции манипулирования данными.

Файловая структура системы LINUX

В операционной системе LINUX файлами считаются обычные файлы, каталоги, а также специальные файлы, соответствующие периферийным устройствам (каждое устройство представляется в виде файла). Доступ ко всем видам файлам однотипный. Такой подход обеспечивает независимость программы пользователя от особенностей ввода/вывода на конкретное внешнее устройство.

Файловая структура LINUX имеет иерархическую древовидную структуру. В корневом каталоге размещаются другие каталоги и файлы, включая 5 основных каталогов:

bin - большинство выполняемых командных программ и shell-процедур;

tmp - временные файлы;

usr - каталоги пользователей;

etc - административные утилиты и файлы;

dev - специальные файлы, представляющие периферийные устройства.

Текущий каталог - это каталог, в котором в данный момент находится пользователь. При наличии прав доступа пользователь может перейти в любой каталог.

Текущий каталог обозначается точкой (.), родительский каталог, которому принадлежит текущий, обозначается двумя точками (..).

Полное имя файла может включать имена каталогов, включая корневой, разделенных косой чертой, например: /home/student/file.txt. Первая косая черта обозначает корневой каталог, и поиск файла будет начинаться с него, а затем в каталоге home, затем в каталоге student.

Один файл можно сделать принадлежащим нескольким каталогам. Для этого использует- ся команда **ln (link)**:

ln <имя 1-го файла> <имя 2-го файла>

Имя 1-го файла - это полное имя файла, с которым устанавливается связь; имя 2-го файла - это полное имя файла в новом каталоге, где будет использоваться эта связь. Новое имя может не отличаться от старого. Каждый файл может иметь несколько связей, то есть он может использоваться в разных каталогах под разными именами. Команда **ln** с аргументом -s создает символическую связь:

ln -s <имя 1-го файла> <имя 2-го файла>

Здесь имя 2-го файла является именем символической связи. Символическая связь является особым видом файла, в котором хранится имя файла, на который эта связь ссылается. LINUX работает с символической связью не так, как с обычным файлом - например, при выводе на экран содержимого символической связи появятся данные файла, на который эта связь ссылается.

В LINUX различаются 3 уровня доступа к файлам и каталогам:

1) доступ владельца файла;

2) доступ группы пользователей, к которой принадлежит владелец файла;

3) остальные пользователи.

Для каждого уровня существуют свои байты атрибутов, значение которых расшифровывается следующим образом:

r – разрешение на чтение;

w – разрешение на запись;

x – разрешение на выполнение;

- – отсутствие разрешения.

Первый символ байта атрибутов определяет тип файла и может принимать следующие значения:

- – обычный файл;

d – каталог;

l – символическая связь;

в – блок-ориентированный файл, который соответствует накопителям на магнитных дисках;

с – байт-ориентированный файл, который соответствует принтеру или терминалу.

В домашнем каталоге пользователь имеет полный доступ к файлам (READ, WRITE, EXECUTE; r, w, x).

Атрибуты файла можно просмотреть командой ls -l и они представляются в следующем формате:

d rwx rwx rwx

| | | |

| | | | Доступ для остальных пользователей
| | | Доступ к файлу для членов группы
| | Доступ к файлу владельца
| Тип файла (директория)

Пример. Командой **ls -l** получим листинг содержимого текущей директории student:

- rwx --- --- 2 student 100 Mar 10 10:30 file_1
- rwx --- r-- 1 adm 200 May 20 11:15 file_2
- rwx --- r-- 1 student 100 May 20 12:50 file_3

После байтов атрибутов на экран выводится следующая информация о файле:

- число связей файла;
- имя владельца файла;
- размер файла в байтах;
- дата создания файла (или модификации);
- время;
- имя файла.

Атрибуты файла и доступ к нему, можно изменить командой **chmod <коды защиты> <имя файла>.**

Коды защиты могут быть заданы в числовом или символьном виде. Для символьного кода используются:

(+) - добавить права доступа;
(-) - отменить права доступа;
r,w,x - доступ на чтение, запись, выполнение;
u,g,o - владельца, группы, остальных.

Коды защиты в числовом виде могут быть заданы в восьмеричной форме. Для контроля установленного доступа к своему файлу после каждого изменения кода защиты нужно проверять свои действия с помощью команды **ls -l.**

Примеры:

chmod g+rw,o+r <файл> - установка атрибутов чтения и записи для группы и чтения для всех остальных пользователей;

ls -l <файл> - чтение атрибутов файла;

chmod o-w <файл> - отмена атрибута записи у остальных пользователей;

>letter - создание файла letter. Символ > используется как для переадресации, так и для создания файла;

cat - вывод содержимого файла;

cat <файл1> <файл2> > <файл3> - конкатенация файлов (объединение);

mv <старое_имя> <новое_имя> - переименование файла;

mv <файл> <directory> - перемещение файла в указанную директорию;

rm <файл> - удаление файла;

cp <файл1> <файл2> - копирование файла с переименованием;

mkdir <каталог> - создание каталога;

rm <каталог> - удаление каталога;

ls [acdfgilqrstv CFR] <каталог|файл> - вывод содержимого. Значения аргументов:

- l – вывести всю информацию о файле;
- t – сортировка по времени модификации файлов;
- a – в список включаются все файлы, в том числе и те, которые начинаются с точки;
- s – размеры файлов указываются в блоках;
- d – вывести имя самого каталога, но не содержимое;
- r – сортировка строк вывода;
- i – указать идентификационный номер каждого файла;
- v – сортировка файлов по времени последнего доступа;
- q – непечатаемые символы заменить на знак ?;
- c – использовать время создания файла при сортировке;
- g – то же что -l, но с указанием имени группы пользователей;
- f – вывод содержимого всех указанных каталогов, отменяет флаги -l, -t, -s, -r и активизирует флаг -a;
- c – вывод элементов каталога в несколько столбцов;
- F – добавление к имени каталога символа / и символа * к имени файла, для которых разрешено выполнение;
- R – рекурсивный вывод содержимого подкаталогов заданного каталога.

cd <каталог> - переход в каталог. Если параметры не указаны, то происходит переход в домашний каталог пользователя.

pwd - вывод имени текущего каталога;

grep [-vcilns] [шаблон поиска] <файл> - поиск файлов с указанием или без указания шаблона.

Значение ключей:

- v – выводятся строки, не содержащие шаблон поиска;
- c – выводится число строк, содержащих или не содержащих шаблон;
- i – регистронезависимый поиск;
- l – выводятся только имена файлов, содержащие указанный шаблон;

- n – перенумерация выводимых строк;
- s – формируется только код завершения.

Примеры.

1. Напечатать имена всех файлов текущего каталога, содержащих последовательность "student" и имеющих расширение .txt:

grep -l student *.txt.

2. Определить имя пользователя, входящего в ОС LINUX с терминала tty23:

who | grep tty23.

Порядок выполнения работы

1. Ознакомиться с файловой структурой ОС LINUX. Изучить команды работы с файлами.
2. Используя команды ОС LINUX, создать два текстовых файла.
3. Полученные файлы объединить в один файл и его содержимое просмотреть на экране.
4. Создать новую директорию и переместить в нее полученные файлы.
5. Вывести полную информацию обо всех файлах и проанализировать уровни доступа.
6. Добавить для всех трех файлов право выполнения членам группы и остальным пользователям.
7. Просмотреть атрибуты файлов.
8. Создать еще один каталог.
9. Установить дополнительную связь объединенного файла с новым каталогом, но под другим именем.
10. Создать символическую связь.
11. Сделать текущим новый каталог и вывести на экран расширенный список информации о его файлах.
12. Произвести поиск заданной последовательности символов в файлах текущей директории и получить перечень соответствующих файлов.
13. Получить информацию об активных процессах и имена других пользователей.
14. Сдать отчет о работе и удалить свои файлы и каталоги.
15. Выйти из системы.

Контрольные вопросы

1. Что считается файлами в ОС LINUX?
2. Объясните назначение связей с файлами и способы их создания.
3. Что определяет атрибуты файлов и каким образом их можно просмотреть и изменить?
4. Какие методы создания и удаления файлов, каталогов Вы знаете?
5. В чем заключается поиск по шаблону?
6. Какой командой можно получить список работающих пользователей и сохранить его в файле?

Список использованных источников

1. Колисниченко Д. Н. Linux. От новичка к профессионалу. - СПб.: БХВ-Петербург, 2010. - 780 с.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 5. «ФАЙЛОВАЯ СИСТЕМА ОС RED HAT LINUX»

Теоретическая часть

Файловая система ОС Red Hat Linux устроена таким образом, что все ресурсы представлены единообразно (в виде файлов). Такой подход позволяет обеспечить универсальный интерфейс доступа к любым ресурсам: от физических устройств до процессов, выполняющихся в системе.

Для обеспечения единообразного доступа к файлам их прежде всего необходимо классифицировать. В Red Hat Linux это сделано следующим образом:

обычные (regular) файлы - текстовые, исполняемые, графические и пр. файлы, создаваемые пользователями и прикладными программами;

каталоги (directories) - именованные группы файлов и вложенных каталогов;

файлы устройств (devices) - соответствуют присутствующим в системе реальным (жесткие диски, принтеры, мыши, ЦП и т.д.) и т.н. псевдоустройствам (например, /dev/null). Файлы устройств представляют символьные (последовательного доступа) и блочные (произвольного доступа) устройства. К первым относят, например, параллельные и последовательные порты, ко вторым - жесткие диски;

специальные файлы - сокеты (sockets) и именованные каналы (named pipes), которые предназначены для обмена информацией между процессами;

символьные ссылки (symlinks) - именованные указатели на физические файлы (аналог ярлыков ОС Windows), содержащие только путь к некоторому файлу. Символьные ссылки могут указывать на файлы, хранящиеся как локальных, так и в сетевых каталогах.

Символьные ссылки (или "мягкие") не нужно путать с "жесткими", которые указывают на inode файла. Inode (идентификатор узла) - это уникальный числовой идентификатор узла (файла или каталога) файловой системы, по которому и осуществляется доступ к нему. Символьное имя файла ориентировано на пользовательское восприятие, так как для человека-оператора проще запомнить осмысленные имена файлов (например: report.txt, myfoto.jpg и т.п.), чем абстрактные числовые значения.

Каталоги Red Hat Linux

Все файлы упорядочены по каталогам. Структура и назначение каждого из каталогов, созданных на этапе установке, предопределены, хотя и могут быть изменены суперпользователем.

Файловая система имеет иерархическую структуру и начинается от корневого каталога (/). Его подкаталогами являются:

/bin - исполняемые файлы общего назначения;

/boot - содержит образ загружаемого ядра;

/dev - файлы устройств;

/etc - конфигурационные файлы общего пользования;

/home - домашние каталоги пользователей, включая программы и файлы личных предпочтений;

/lib - общесистемные библиотеки;

/mnt - каталог монтирования внешних файловых систем;

/proc - виртуальная файловая система для чтения информации о процессах;

/root - домашний каталог суперпользователя;

/sbin - программы системного администрирования;

/tmp - каталог для хранения временной информации;

/usr - каталог пользовательских прикладных программ со всеми их исполнимыми и конфигурационными файлами;

/var - каталог для хранения часто изменяющихся файлов, например, спулера печати, различных лог-файлов, почтовых сообщений и т.п.

/lost+found - каталог для нарушенных фрагментов файлов, обнаруженных в результате проверки файловой системы после сбоя.

Такая структура типична для большинства дистрибутивов Linux, но ОС может иметь и другие каталоги, например, /opt - для дополнительных компонентов, /selinux - расширение системы безопасности и т.п.

Именование файлов и каталогов

Файловая система Red Hat Linux поддерживает "длинные" имена, содержащие символы латиницы, национальных алфавитов, знаки пунктуации и спецсимволы. Абсолютно запрещенными к использованию в имени являются прямой и обратный слэши (/ и \). Максимальное количество символов в имени - 255. Понятие "расширения файла" в Unix-системах отсутствует как таковое, поэтому в имени может быть несколько частей, разделенных точка-

ми. Все имена - *регистрозависимые*.

Файлы и каталоги, названия которых начинаются с точки (dot-файлы), являются аналогами "скрытых" файлов MS-DOS, то есть не отображаются при просмотре содержимого файловой системы.

Для быстрого доступа к файлам в оболочке имеются несколько переменных окружения, хранящих соответствующие пути. Это, например, переменная $HOME, в которой содержится пути к домашнему каталогу текущего пользователя. Таким образом, выполнение команд

```
[usr1@localhost var]$ cd /home/usr1
```

и

```
[usr1@localhost var]$ cd $HOME
```

приведут к одному результату - переходу в домашний каталог пользователя usr1. Более того, в оболочке определен псевдоним для домашнего каталога - символ ~ (тильда) можно использовать аналогично $HOME. Например:

```
[usr1@localhost var]$ cd ~
[usr1@localhost ~]$ pwd
/home/usr1
[usr1@localhost var]$
```

Команды управления файловой системой

Управление файловой системой производится при помощи различных команд, реализующих операции по созданию, чтению, копированию, переименованию/перемещению, изменению и удалению файлов и каталогов.

Основными командами для выполнения файловых операций являются: pwd, ls, cp, mv, dir, rm, cd, rmdir, mvdir, mkdir, ln. Информация об их назначении и параметрах доступна в справках man и info.

Файлы в Red Hat Linux могут быть созданы как результаты работы прикладных программ и иметь при этом определенный формат (например, графические файлы, созданные редактором GIMP) или могут быть созданы пользователем путем ввода информации с клавиатуры, например:

```
aag@stilo:~> cat > f1
Hello, world! // нажатие Ctrl+D завершает ввод команд
aag@stilo:~>
```

Также файлы могут быть созданы путем перенаправления вывода команды со стандартного потока:

```
[root@localhost aag]# echo "Hello, World!" > f1
```

В данных примерах символ ">" - это команда перенаправления стандартных потоков ввода/вывода, встроенная в оболочку. В первом случае она получает информацию со стандартного потока ввода (клавиатура) и по окончании ввода (Ctrl+D) отправляет ее в файл. Во втором - принимает строку, переданную командой echo, и также отправляет ее в файл. Если файл отсутствует, то он будет создан, если имеется, то будет **перезаписан**. Для **добавления** информации в файл следует использовать команду ">>".

Задания для самостоятельной работы

1. Войти в систему под собственной учетной записью.
2. Вывести на экран список файлов текущего каталога в краткой и расширенной форме.
3. Переместиться в каталог /.
4. Сохранить в файле $HOME/filelist.lst список каталогов в каталоге /.
5. Вернуться в домашний каталог и вывести рекурсивный список всех (в т.ч. и скрытых) файлов и каталогов.
6. В домашнем каталоге создать подкаталоги src, dst и temp.
7. В каталоге src создать текстовый файл f1 произвольного содержания.
8. В каталог src скопировать файлы user.txt, root. txt и stud.txt, созданные в лабораторной работе № 2. Проверить, все ли файлы удалось скопировать.
9. В каталоге dst создать «жесткие» ссылки на все файлы из каталога src .
10. В домашнем каталоге создать «мягкие» ссылки на файлы из каталога src .
11. Вывести рекурсивно расширенную информацию о содержимом домашнего каталога. Обратить внимание на поле размера для физических файлов и ссылок.
12. Из домашнего каталога выполнить команды:

    cat /src/f1;
    cat /dst/f1;
    cat /f1.

    Запомнить результаты выполнения команд.
13. Переместить файл f1 из каталога src в каталог temp и повторить п.12.
14. Удалить файл f1 и повторить п.12.

15. Сравнить результаты выполнения пп 12, 13 и 14.
16. Удалить все файлы, имеющие в названии txt из каталога dst.
17. Удалить каталог dst.
18. Переместить каталог temp в src.
19. Рекурсивно удалить каталог src.
20. Завершить сеанс.

Контрольные вопросы

1. Классификация файлов в Red Hat Linux.
2. Что такое символьные ссылки?
3. Назовите минимум пять подкаталогов корневого каталога файловой системы.
4. Что такое регистрозависимые имена?
5. Назовите основные команды для выполнения файловых операций.
6. Какие существуют способы создания файлов в Red Hat Linux?

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 6. «ПРОЦЕССЫ. ДОСТУП ПРОЦЕССОВ К ФАЙЛОВОЙ СИСТЕМЕ»

Теоретическая часть

Под процессами во всех Unix-подобных системах подразумевается любая независимо выполняющаяся программа со всеми используемыми ей ресурсами. Процесс тесно связан с такими понятиями, как учетные записи, права доступа и файловая система. Эта связь неразрывна и рекурсивна:

— пользователь (реальный или виртуальный) запускает процессы, порождающие файлы;

— права доступа процесса соответствуют правам доступа запустившего его пользователя;

— порожденные процессом файлы получают права, соответствующие правам процесса;

— права пользователя определены параметрами его учетной записи.

Каждый процесс уникален. Для идентификации процесса используется числовое значение - идентификатор процесса (PID, Process Identificator). Для каждого процесса известен владелец (пользователь, запустивший процесс). Если процесс создан реальным пользователем, то он привязан к терминалу, из которого был запушен. Для виртуальных (системных) пользователей такой ассоциации не предусмотрено. Также каждый процесс связан с родительским процессом по его идентификатору (PPID, Parent PID). В каждый момент времени системе известно состояние процесса - степень его исполнения. Процесс может быть:

— выполняемым в текущий момент (R, Runned);

— находящимся в режиме ожидания (S, Suspended);

— прерванным (T, Terminated), например, при использовании клавиш Ctrl+Z;

— "зомби" (Z, Zombied) - завершившимся, но от которого родительский процесс еще не принял сигнала завершения. Спустя некоторое время "зомби" завершаются окончательно и освобождают ресурсы;

— зависшим, или в состоянии непрерывного ожидания (uninterruptible sleep). Такой процесс не реагирует на какие-либо сигналы и может быть снят только перезагрузкой системы.

Еще одной характеристикой процесса является уровень приоритета (NI, NIce value, "степень дружественности"). Он влия-

ет на количество системных ресурсов, выделяемых процессу.

Основными командами для получения сведений о выполняемых процессах являются ps и top. Фрагмент вывода сведений командой ps с параметрами a (расширенный вывод), u (с указанием UID), x (в т.ч для виртуальных пользователей):

```
aag@stilo:~> ps aux
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
root 1 0.0 0.0 744 284 ? Ss 14:27 0:00 init [5]
root 2 0.0 0.0 0 0 ? S< 14:27 0:00 [kthreadd]
root 3 0.0 0.0 0 0 ? S< 14:27 0:00 [migration/0]
root 4 0.0 0.0 0 0 ? SN 14:27 0:00 [ksoftirqd/0]
root 5 0.0 0.0 0 0 ? S< 14:27 0:00 [events/0]
root 6 0.0 0.0 0 0 ? S< 14:27 0:00 [khelper]
root 25 0.0 0.0 0 0 ? S< 14:27 0:00 [kblockd/0]
...
root 141 0.0 0.0 0 0 ? S< 14:27 0:00 [kswapd0]
root 142 0.0 0.0 0 0 ? S< 14:27 0:00 [aio/0]
root 367 0.0 0.0 0 0 ? S< 14:27 0:00 [kpsmoused]
root 377 0.0 0.0 0 0 ? S< 14:27 0:00 [kondemand/0]
...
root 991 0.0 0.0 0 0 ? D< 14:28 0:01 [kjournald]
root 1682 0.0 0.0 0 0 ? S< 14:28 0:01 [ipw2200/0]
root 1694 0.0 0.0 0 0 ? S< 14:28 0:00 [khpsbpkt]

wwwrun 3323 0.0 2.4 104548 12804 ? S 14:28 0:00 /usr/sbin/httpd
wwwrun 3324 0.0 2.4 104540 12816 ? S 14:28 0:00 /usr/sbin/httpd
```

Большей гибкостью и универсальностью по сравнению с командой ps обладает команда top. Она позволяет не только получить информацию о процессах, но и выполнять мониторинг через заданные интервалы времени. Так же эта команда позволяет управлять процессами, объединяя возможности команд jobs, nice, fg и kill.

Управление процессами

Пользователь может управлять только теми процессами, владельцем которых является. Но суперпользователь может управлять всеми процессами.

Управление запущенными процессами сводится к приоста-

новке выполнения, изменению приоритета и принудительному завершению.

Приостановить выполнение активного процесса можно сочетанием клавиш Ctrl+Z. Для продолжения его работы следует использовать команду fg. Если имеется несколько приостановленных процессов, то в качестве параметра команды fg необходимо указать порядковый номер задания в текущей оболочке, (не путать с PID), работу которого нужно продолжить. Узнать номер задания можно командой jobs.

Изменение приоритета процесса возникает при необходимости перераспределения ресурсов системы. Значения уровня приоритета (nice value) изменяется от -20 (наименьшая "дружественность", высший приоритет) до +20 (низший приоритет). Все пользовательские (и большинство системных) процессы запускаются с равным приоритетом (nice value = 0).

Для понижения приоритета ранее запущенной задачи используется команда renice с указанием уровня и PID задачи:

[aag@localhost ~]$ renice --7 20117.

Пользователь имеет право *понижать* приоритет *собственных* задач. Повышать уровень приоритета *любой задачи* может только суперпользователь.

Гораздо чаще, чем изменение приоритета, возникает необходимость принудительного завершения (снятия) процесса. Такая ситуация возникает, например, когда процесс "зависает", то есть перестает воспринимать нажатия клавиш и не отвечает на системные события. Для снятия "зависшей" программы предназначена команда kill, которая передает ей один из сигналов завершения. Список сигналов доступен по команде kill -l, а их подробное описание - по команде man 7 signal. Здесь же отметим, что без явного указания имени (или номера), процессу будет передан сигнал SIGTERM (номер 15), предписывающий *по возможности* корректно, с сохранением информации, завершить работу.

Примеры использования команды:

Вызов со значением сигнала по умолчанию (SIGTERM):

```
[aag@localhost ~]$ find / *.html
[aag@localhost ~]$ ps
PID TTY TIME CMD
2663 pts/1 00:00:00 bash
20712 pts/1 00:00:00 find
20762 pts/1 00:00:00 ps
[aag@localhost ~]$ kill 20712
```

Явное указание номера сигнала:

[aag@localhost ~]$ kill -15 20712

Явное указание имени сигнала (номер 9, SIGKILL, требующий немедленного завершения работы программы):

[aag@localhost ~]$ kill -SIGKILL 20712

Задания для самостоятельной работы

1. Войти в систему с собственной учетной записью.
2. Получить справку о команде ps.
3. Командой ps вывести краткую информацию о выполняющихся процессах в текущем терминале и определить PID текущей оболочки.
4. Получить подробную информацию о загруженных процессах и выяснить, какой из них использует максимальный объем памяти, а какой - максимально загружает процессор.
5. Из таблицы, полученной в п.4, выяснить, какой PID имеет процесс init и от чьего имени он запущен.
6. Открыть новый сеанс с собственной учетной записью в tty2 и запустить в нем файловый менеджер MC.
7. Вернуться в tty1 и снова просмотреть список процессов. Определить PID MC, запущенного от вашего имени.
8. Повторить п.6 для пользователей root и stud соответственно в tty3 и tty4.
9. Вернуться в tty1 и определить PID MC, запущенного от имени root и stud.
10. Командой kill снять все процессы MC.
11. Перейти в tty3 (сеанс root) и повторить п.10. Чем можно объяснить различия в результатах выполнения?
12. В tty1 выполнить команду top. Сравнить ее возможности с возможностями ps.
13. Используя top или ps определить, какие процессы порождены (поле PPID) процессом init (PID=1).
14. Завершить сеансы в tty3 и tty4.
15. В tty1 запустить поиск всех файлов .html от каталога /. Приостановить этот процесс (Ctrl+Z). Запустить команду man bash и приостановить ее выполнение.
16. Командой jobs определить номера задач, запущенных в п. 15.
17. Командой fg продолжить выполнение man bash.
18. Принудительно (kill) завершить команду find.
19. Завершить все открытые сеансы.

Контрольные вопросы

1. Что подразумевается под процессами в Unix-подобных системах?
2. Каким может быть процесс?
3. На что влияет уровень приоритета процесса?
4. Основными командами для получения сведений о выполняемых процессах являются…
5. Что такое изменение приоритета процесса?

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 7. «ИЗУЧЕНИЕ МЕТОДОВ СОЗДАНИЯ И ВЫПОЛНЕНИЯ КОМАНДНЫХ ФАЙЛОВ НА ЯЗЫКЕ SHELL – ИНТЕРПРЕТАТОРА»

Теоретическая часть

В предыдущих лабораторных работах взаимодействие с командным интерпретатором Shell осуществлялось с помощью командной строки. Однако Shell является также и языком программирования, который применяется для написания командных файлов (shell - файлов). Командные файлы также называются скриптами и сценариями. Shell - файл содержит одну или несколько выполняемых команд (процедур), а имя файла в этом случае используется как имя команды.

Переменные командного интерпретатора

Для обозначения переменных Shell используется последовательность букв, цифр и символов подчеркивания; переменные не могут начинаться с цифры. Присваивание значений переменным проводится с использованием знака =, например, PS2 = '<' . Для обращения к значению переменной перед ее именем ставится знак $. Их можно разделить на следующие группы:

- позиционные переменные вида $n, где n - целое число;
- простые переменные, значения которых может задавать пользователь или они могут устанавливаться интерпретатором;
- специальные переменные # ? - ! $ устанавливаются интерпретатором и позволяют получить информацию о числе позиционных переменных, коде завершения последней команды, идентификационном номере текущего и фонового процессов, о текущих флагах интерпретатора Shell.

Простые переменные. Shell присваивает значения переменным:

```
z=1000
x= $z
echo $x
1000
```

Здесь переменной x присвоено значение z.

Позиционные переменные. Переменные вида $n, где n - целое число, используются для идентификации позиций элементов в командной строке с помощью номеров, начиная с нуля. Напри-

мер, в командной строке

cat text_1 text_2...text_9

аргументы идентифицируются параметрами $1...$9. Для имени команды всегда используется S0. В данном случае $0 - это cat, $1 - text_1, $2 - text_2 и т.д. Для присваивания значений позиционным переменным используется команда set, например:

set arg_1 arg_2... arg_9.

Здесь $1 присваивается значение аргумента arg_1, $2 - arg_2 и т.д.

Для доступа к аргументам используется команда echo, например:

echo $1 $2 $9

arg_1 arg_2 arg_9.

Для получения информации обо всех аргументах (включая последний) используют метасимвол *. Пример:

echo $*

arg_2 arg_3 ... arg_10 arg_11 arg_12.

С помощью позиционных переменных Shell можно сохранить имя команды и ее аргументы. При выполнении команды интерпретатор Shell должен передать ей аргументы, порядок которых может регулироваться также с помощью позиционных переменных.

Специальные переменные. Переменные - ? # $! устанавливаются только Shell. Они позволяют с помощью команды echo получить следующую информацию:

- – текущие флаги интерпретатора (установка флагов может быть изменена командой set);

– число аргументов, которое было сохранено интерпретатором при выполнении какой-либо команды;

? – код возврата последней выполняемой команды;

$ – числовой идентификатор текущего процесса PID;

! – PID последнего фонового процесса.

Арифметические операции

Команда **expr** (express -- выражать) вычисляет выражение expression и записывает результат в стандартный вывод. Элементы выражения разделяются пробелами. Символы, имеющие специальный смысл в командном языке, необходимо экранировать. Для этого строки, содержащие специальные символы, заключают в апострофы. Используя команду expr, можно выполнять сложение, вычитание, умножение, деление, взятие остатка, сопоставление символов и т. д.

Пример. Сложение, вычитание:

```
b=190
a=` expr 200 - $b.
```

Умножение *, деление /, взятие остатка %:

```
d=` expr $a + 125 "*" 10`
c=` expr $d % 13`.
```

Здесь знак умножения заключается в двойные кавычки, чтобы интерпретатор не воспринимал его как метасимвол.

Сопоставление символов с указанием числа совпадающих символов:

```
concur=` expr "abcdefgh" : "abcde"`
echo $concur
5.
```

Операция сопоставления обозначается двоеточием (:).

Подсчет числа символов в цепочках символов. Операция выполняется с использованием функции length в команде expr:

```
chain="The program is written in Assembler"
str=` expr length "$chain"`
Echo $str
35.
```

Встроенные команды

Встроенные команды являются частью интерпретатора и не требуют для своего выполнения проведения последовательного поиска файла команды и создания новых процессов:

cd [dir] - назначение текущего каталога;

exec [cmd [arg...]] <имя файла> - выполнение команды, заданной аргументами cmd и arg, путем вызова соответствующего выполняемого файла.

umask [-o | -s] [nnn] - устанавливает маску создания файла (маску режимов доступа создаваемого файла, равную восьмеричному числу nnn: 3 восьмеричных цифры для пользователя, группы и других). Если аргумент nnn отсутствует, то команда сообщает текущее значение маски. При наличии флага -o маска выводится в восьмеричном виде, при наличии флага -s - в символьном представлении;

set, unset - режим работы интерпретатора, присваивание значений параметрам;

eval [-arg] - вычисление и выполнение команды;

sh <filename.sh> - выполнение командного файла filename.sh;

exit [n] - приводит к прекращению выполнения программы,

возвращает код возврата, равный нулю, в вызывающую программу;

trap [cmd] [cond] - перехват сигналов прерывания,

где: cmd - выполняемая команда;

cond=0 или EXIT - в этом случае команда cmd выполняется при завершении интерпретатора;

cond=ERR - команда cmd выполняется при обнаружении ошибки;

cond - символьное или числовое обозначение сигнала, в этом случае команда cmd выполняется при приходе этого сигнала;

export [name [=word]...] - включение в среду. Команда export объявляет, что переменные name будут включаться в среду всех вызываемых впоследствии команд;

wait [n] - ожидание завершения процесса. Команда без аргументов ожидает завершения процессов, запущенных синхронно. Если указан числовой аргумент n, то wait ожидает фоновый процесс с номером n;

read name - команда вводит строку со стандартного ввода и присваивает прочитанные слова переменным, заданным аргументами name.

Пример. Пусть имеется shell-файл data, содержащий две команды:

```
echo -n "Please write down your name:"
read name
```

Если вызвать файл на выполнение, введя его имя, то на экране появится сообщение:

Please write down your name:

Программа ожидает ввода с клавиатуры. После завершения ввода команда выполнится.

Управление программами

Команды **true** и **false** служат для установления требуемого кода завершения процесса:

true - успешное завершение, код завершения 0;

false - неуспешное завершение, код может иметь несколько значений, с помощью которых определяется причина неуспешного завершения.

Коды завершения команд используются для принятия решения о дальнейших действиях в операторах цикла **while** и **until** и в условном операторе **if**. Многие команды LINUX вырабатывают код завершения только для поддержки этих операторов.

Условный оператор if проверяет значение выражения. Если оно равно true, Shell выполняет следующий за **if** оператор, если false, то следующий оператор пропускается. Формат оператора **if**:

if <условие>
then
list1
else
list2
fi

Команда **test** (проверить) используется с условным оператором **if** и операторами циклов. Действия при этом зависят от кода возврата **test**. **Test** проводит анализ файлов, числовых значений, цепочек символов. Нулевой код выдается, если при проверке результат положителен, ненулевой код при отрицательном результате проверки.

В случае анализа файлов синтаксис команды следующий:

test [-rwfds] file

где -r – файл существует и его можно прочитать (код завершения 0);

-w – файл существует и в него можно записывать;

-f – файл существует и не является каталогом;

-d – файл существует и является каталогом;

-s – размер файла отличен от нуля.

При анализе числовых значений команда **test** проверяет, истинно ли данное отношение. Сравнение выполняется в формате:

-eq a = B
-ne a <> B
-ge a >= B
-le a <= B
-gt a > B
-lt a < B

Кроме команды **test** имеются еще некоторые средства для проверки:

! - операция отрицания инвертирует значение выражения, например, выражение **if test true** эквивалентно выражению **if test ! false**;

o - двуместная операция "ИЛИ" (or) дает значение true, если один из операндов имеет значение true;

a - двуместная операция "И" (and) дает значение true, если оба операнда имеют значение true.

Циклы

Команда **while** (пока) формирует циклы, которые выполняются до тех пор, пока команда **while** определяет значение следующего за ним выражения как true или false. Формат оператора цикла с условием **while** true:

while list1
do
list2
done

Здесь list1 и list2 - списки команд. **While** проверяет код возврата списка команд, стоящих после **while**, и если его значение равно 0, то выполняются команды, стоящие между **do** и **done**. Оператор цикла с условием **while** false имеет формат:

until list1
do
list2
done

В отличие от предыдущего случая условием выполнения команд между **do** и **done** является ненулевое значение возврата. Программный цикл может быть размещен внутри другого цикла (вложенный цикл). Оператор **break** прерывает ближайший к нему цикл. Если в программу ввести оператор **break** С уровнем 2 (**break 2**), то это обеспечит выход за пределы двух циклов и завершение программы.

Оператор **continue** передает управление ближайшему в цикле оператору **while**.

Оператор цикла с перечислением **for**:

for name in [wordlist]
do
list
done

где name - переменная;
wordlist - последовательность слов;
list - список команд.

Переменная name получает значение первого слова последовательности wordlist, после этого выполняется список команд, стоящий между **do** и **done**. Затем name получает значение второго слова wordlist и снова выполняется список list. Выполнение прекращается после того, как кончится список wordlist.

Команда **case** обеспечивает ветвление по многим направ-

лениям в зависимости от значений аргументов команды. Формат:

```
case <string> in
s1) <list1>;;
s2) <list2>;;
.
.
.
sn) <listn>;;
*) <list>
esac
```

Здесь list1, list2 ... listn - список команд. Производится сравнение шаблона string с шаблонами s1, s2 ... sk ... sn. При совпадении выполняется список команд, стоящий между текущим шаблоном sk и соответствующими знаками ;;. Пример:

```
echo -n 'Please, write down your age'
read age
case $age in
test $age -le 20) echo 'you are so young' ;;
test $age -le 40) echo 'you are still young' ;;
test $age -le 70) echo 'you are too young' ;;
*)echo 'Please, write down once more'
esac
```

В конце текста помещена звездочка * на случай неправильного ввода числа.

Порядок выполнения работы

Составьте и выполните shell - программы, включающей следующие действия:

1. Вывод на экран списка параметров командной строки с указанием номера каждого параметра.
2. Присвоение переменным A, B и c значений 10, 100 и 200, вычисление и вывод результатов по формуле D=(A*2 + B/3)*C.
3. Формирование файла со списком файлов в домашнем каталоге, вывод на экран этого списка в алфавитном порядке и общего количества файлов.
4. Переход в другой каталог, формирование файла с листингом каталога и возвращение в исходный каталог.
5. Запрос и ввод имени пользователя, сравнение с текущим логическим именем пользователя и вывод сообщения: верно/неверно.
6. Запрос и ввод имени файла в текущем каталоге и вывод

сообщения о типе файла.

7. Циклическое чтение системного времени и очистка экрана в заданный момент.

8. Циклический просмотр списка файлов и выдача сообщения при появлении заданного имени в списке.

Контрольные вопросы

1. Какое назначение имеют shell - файлы?
2. Как создать shell - файл и сделать его выполняемым?
3. Какие типы переменных используются в shell - файлах?
4. В чем заключается анализ цепочки символов?
5. Какие встроенные команды используются в shell - файлах?
6. Как производится управление программами?
7. Назовите операторы создания циклов.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 8. «УПРАВЛЕНИЕ ПРАВАМИ ДОСТУПА»

Теоретическая часть

Разграничение прав доступа в ОС Red Hat Linux обусловлено многозадачностью и многопользовательским режимом ОС и призвано повысить безопасность и надежность системы, а также обеспечить защиту конфиденциальной информации.

Каждый файл характеризуется набором атрибутов, определяющих его принадлежность и права доступа. Отношение принадлежности файла определено для:

владельца файла(user) - пользователя, создавшего этот файл;

группы (group) - в состав которой входит владелец;

прочих (other) пользователей.

К правам доступа относятся: чтение (read), изменение (write), исполнение (execute). Понимание этих прав будет различным и зависеть от содержания файла. Наибольшие различия - между обычными (regular) файлами и каталогами. Эти различия приведены в табл. 1.

	Файлы	**Каталоги**
Чтение	Просмотр содержимого файла (например, текста) в соответствующей программе и возможность его копирования	Обзор списка файлов и возможность копирования каталога (в общем случае, вместе со всем содержимым)
Изменение	Редактирование содержимого файла и его копирование, но не удаление или переименование/перемещение	Обеспечивает возможность записи и удаления файлов
Исполнение	Разрешает запуск программ и сценариев оболочки	Разрешает переход в каталог и перемещение по нему

Атрибуты файла могут быть представлены в символьном

или числовом виде. Символьное представление атрибутов - это строка, где последовательно записаны права доступа в следующем виде:

rwxrwxrwx

где каждая тройка символов определяет права на чтение (**r**), запись (**w**) и исполнение (**x**) для соответствующих пользователей (первая тройка - для владельца (**u**ser), вторая - для группы (**g**roup), третья - для прочих (**o**ther)).

Вот пример отображения списка файлов с правами доступа, представленными в символьном виде:

```
aag@stilo:~> dir -L1
итого 2722316
-rw-r--r-- 1 aag users 498444757 Ноя 27 16:15 aag.asoiu.tar.gz
drwxr-xr-x 2 aag users 4096 Июн 1 2007 bin
-rw-r--r-- 1 aag users 26 Фев 20 10:20 description.txt
drwxr-xr-x 5 aag users 4096 Мар 2 20:01 Desktop
drwx------ 2 aag users 4096 Фев 23 09:50 Documents
drwxr-xr-x 4 aag users 4096 Фев 28 00:03 downloads
-rwxrwxr-x 1 aag users 7523 Окт 20 2006 Dz19.jpg
-rw-r--r-- 1 aag users 8336 Фев 24 01:12 httpd.myconf
-rw-r--r-- 1 aag users 20 Фев 25 16:32 index.html
-rw-r--r-- 1 aag users 30296 Фев 23 10:05 logofish.xcf
drwxr-xr-x 2 aag users 4096 Сен 28 22:53 Music
drwxr-xr-x 3 aag users 4096 Дек 3 13:45 Projects
drwxr-xr-x 4 aag users 4096 Фев 26 00:05 public_html
-rw-r--r-- 1 aag users 1088 Фев 20 10:18 readme.txt
drwxr-xr-x 4 aag users 4096 Фев 27 23:41 scrapbook
-rw------- 1 root root 0 Июн 2 2007 session_mm_cli0.sem
```

Обратите внимание на первые символы в записи прав доступа. В приведенном листинге первый символ **d** указывает, что файл является каталогом. Признаком специального символьного и блочного устройств являются символы **c** и **b**, а для каналов (pipes) соответственно **p**.

Числовое представление прав доступа - это трехзначное число, каждая цифра которого определяет (слева направо) права для владельца, группы и прочих. Права определяются как сумма цифр 4 (чтение), 2 (запись) и 1 (исполнение). Таким образом, например файл, разрешенный для чтения и изменения членам группы и только чтение всем прочим, будет иметь следующие атрибуты:

- в символьном виде: *rw-rw-r—*
- в числовом виде: *664*.

Вновь создаваемый файл обычно получает права rw-r--r-- (зависит от установок системы и значения umask). Для изменения атрибутов используется команда chmod, которая может принимать как символьное, так и числовое представление атрибутов в качестве параметра. Ниже приведены примеры использования команды:

aag@stilo:~> dir hello.txt

-rw-r--r-- 1 aag users 17 Map 2 22:32 hello.txt

aag@stilo:~> chmod **go+w** hello.txt // *разрешить запись для группы и прочих*

aag@stilo:~> dir hello.txt

-rw-rw-rw- 1 aag users 17 Map 2 22:32 hello.txt

aag@stilo:~> chmod **ug+x** hello.txt // *разрешить выполнение для владельца и группы*

aag@stilo:~> dir hello.txt

-rwxrwxrw- 1 aag users 17 Map 2 22:32 hello.txt

aag@stilo:~> chmod **a-x** hello.txt // *запретить выполнение для всех (**a** == ugo)*

aag@stilo:~> dir hello.txt

-rw-rw-rw- 1 aag users 17 Map 2 22:32 hello.txt

aag@stilo:~> chmod **go-w** hello.txt // *запретить запись для группы и прочих*

aag@stilo:~> dir hello.txt

-rw-r--r-- 1 aag users 17 Map 2 22:32 hello.txt

aag@stilo:~> chmod **755** hello.txt // *разрешить чтение и выполнение всем и запись владельцу*

aag@stilo:~> dir hello.txt

-rwxr-xr-x 1 aag users 17 Map 2 22:32 hello.txt

aag@stilo:~> chmod **644** hello.txt // *запретить выполнение всем*

aag@stilo:~> dir hello.txt

-rw-r--r-- 1 aag users 17 Map 2 22:32 hello.txt

aag@stilo:~> chmod **711** hello.txt // *разрешить только выполнение для группы и прочих*

aag@stilo:~> dir hello.txt

-rwx--x--x 1 aag users 17 Map 2 22:32 hello.txt

Для смены владельца файла и группы используется команда chown, а для смены группы - команда chgrp.

Задания для самостоятельной работы

1. Войти в систему с собственной учетной записью.

2. Создать в домашнем каталоге 2-3 файла произвольного содержания (имена файлов - u1, u2, u3).

3. Получить развернутый список файлов домашнего каталога и сохранить его в файле listing1.

4. Просмотреть файл listing1, обратив внимание на поля прав доступа, владельца и группы.

5. Повторить п. 2 от имени пользователя root в новом сеансе или по команде su (имена файлов - r1, r2, r3). Завершить сеанс root.

6. Повторить п.3, результат дописать в файл listing1.

7. Открыть файл listing1 и сравнить права доступа для файлов, созданных от вашего имени и от имени суперпользователя.

8. Изменить содержимое файлов, созданных вами и суперпользователем. Сохранить изменения.

9. В tty2 открыть сеанс root.

10. Перейти в каталог /home/ваша_учетная_запись.

11. Изменить права доступа к файлам u1 и r1 следующим образом:

u1: запретить запись для владельца и группы;

r1: разрешить запись для всех.

12. Переключиться в tty1 и изменить содержимое файлов u1 и r1. Сохранить изменения.

13. Перейти в tty2 и изменить владельца файлов u1 и u2 на root,а группу на stud.

14. Из tty1 попробовать изменить файл u2.

15. В tty1 создать файл hello следующего содержания

```
#! /bin/sh
echo Hello, World!
echo -n "I'm "
whoami.
```

16. Выполнить следующие действия и проанализировать результаты:

набрать в командной строке имя файла hello и нажать Enter;

набрать в командной строке sh hello и нажать Enter;

установить для файла hello права на исполнение (x), ввести имя файла в командной строке (./hello) и нажать Enter.

17. Из tty2 создать каталоги /home/shared, home/shared/pub, /home/shared/upload, /home/shared/temp. Установить на них следующие права:

каталог	владелец	группа	права
pub	root	users	775
upload	nobody	users	130
temp	stud	users	777

18. Выполнить копирование, чтение, удаление файлов u1, u2, u3, r1, r2, r3 в каталоги, созданные в п. 17 из сеансов root, stud и вашего. Сравнить и проанализировать результаты.

19. Завершить все сеансы.

Контрольные вопросы

1. Что относят к права доступа?
2. В каком виде могут быть представлены атрибуты файла?
3. Какая команда используется для смены владельца файла и группы?

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 9. «ВЫПОЛНЕНИЕ СЦЕНАРИЕВ В UNIX»

Теоретическая часть

Сценарий: Вход в систему и завершение сеанса

Для входа в систему необходимо ввести логин и пароль пользователя. При этом символы вводимого пароля не отображаются.

```
Welcome to desktop / tty1
desktop login: user
Password:
Last login: Wed Dec 07 00:20:09 2006 from tty1
user@desktop ~ $
```

При входе в систему в графическом режиме (через X11), пользователю вместо командной строки предоставляется графический десктоп.

Для завершения сеанса работы в командной строке необходимо ввести команду exit:

```
user@desktop ~ $ exit.
```

Сценарий: Изучение базовых прав доступа

Для просмотра прав доступа можно использовать команду ls с ключом вывода расширенной информации: **ls -l**.

```
user@desktop ~ $ ls -l
drwxr-xr-x  1 user users  22 Дек 19 11:18 test/
-rw-r--r--  1 user users  90 Сен 19 00:20 test.txt
```

Особый интерес представляют: первая колонка (права доступа), третья и четвёртая – владелец и группа владельцев соответственно.

Рассмотрим исполняемый файл **ls -l /bin/bash**.

```
user@desktop ~ $ ls -l /bin/bash
-rwxr-xr-x  1 root root 746544 Дек 21 13:40 /bin/bash*
```

Исполняемые файлы в UNIX определяются наличием специального бита прав доступа.

Для директории права на исполнения трактуются по-другому. Рассмотрим права директории /tmp **ls -ld /tmp**:

```
user@desktop ~ $ ls -ld /tmp
drwxrwxrwt  26 root root 5168 Дек 22 20:04 /tmp/
```

Директория имеет дополнительный sticky-bit, определяю-

щий права на создание и удаление файлов в директории.

Изменение файла с недостатком прав приводит к ошибке доступа. Например, команда: **rm /bin/bash**.

```
user@desktop ~ $ rm /bin/bash
rm: удалить защищенный от записи обычный файл `/bin/bash'? y
rm: невозможно удалить `/bin/bash': Permission denied.
```

Для всех файлов, на которые данный пользователь не имеет прав записи, команда rm выведет предупреждение об удалении.

Изменение прав доступа производится с помощью команды chmod. Для задания файлу прав только для чтения воспользуемся командой: **chmod a=r test.txt**.

Для лишения всех прав группы владельцев и остальных пользователей воспользуемся командой: **chmod go-rwx test.txt**.

Сценарий: Переход в режим суперпользователя

Некоторые исполняемые программы обладают специальным suid-битом, например, программа passwd. Рассмотрим права доступа к этому исполняемому файлу: **ls -l /bin/passwd**.

```
user@desktop ~ $ ls -l /bin/passwd
-rws--x--x 1 root root 28660 Янв 8 13:05 /bin/passwd*
```

Запускаются suid-программы от имени владельца файла. В этом можно убедиться, если запустить команду **passwd**, а затем на другом терминале сделать **ps aux | grep passwd**:

```
user@desktop ~ $ passwd
Changing password for user
(current) UNIX password:
user@desktop ~ $ ps aux | grep passwd
root 12937 0.0 0.1 3228 1012 pts/2 S+ 23:28 0:00 passwd
user 12989 0.0 0.1 2740 748 pts/3 R+ 23:28 0:00 grep passwd
```

Для длительной работы в режиме суперпользователя обычно используют команду su. Используем параметр -, чтобы проинициализировать окружение суперпользователя: **su -**. Для того, чтобы перейти в режим суперпользователя, необходимо знать его пароль.

```
user@desktop ~ $ su -
Password:
desktop ~ #
```

При этом запускается новая командная оболочка, уже с новыми привелегиями.

Для завершения сеанса суперпользователя необходимо

воспользоваться командой **exit**:

```
desktop ~ # exit
logout
user@desktop ~ $
```

Сценарий: Изучение базы данных пользователей

Данные о зарегистрированных в системе пользователях хранятся в файле /etc/passwd. Рассмотрим его содержимое **cat /etc/passwd**:

```
user@desktop ~ $ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/false
daemon:x:2:2:daemon:/sbin:/bin/false
adm:x:3:4:adm:/var/adm:/bin/false
```

Аналогичным образом данные о группах хранятся в файле /etc/group. Рассмотрим его содержимое **cat /etc/group**:

```
user@desktop ~ $ cat /etc/group
root:x:0:root
bin:x:1:root,bin,daemon
daemon:x:2:root,bin,daemon
sys:x:3:root,bin,adm.
```

Сценарий: Добавление и удаление пользователей

Изменения учётных записей пользователей доступны только суперпользователю.

Для добавления новых пользователей используется команда useradd. При этом в качестве параметров можно указать домашнюю директорию и командную оболочку пользователя: **useradd testuser -d /home/users/testuser -s /bin/sh**, результат можно увидеть следующим образом: **cat /etc/passwd | grep testuser**.

```
desktop ~ # cat /etc/passwd
testuser:x:1003:1003::/home/users/testuser:/bin/sh
```

С помощью команды passwd можно задать пароль для нового пользователя: **passwd testuser**.

```
desktop ~ # passwd testuser
New UNIX password:
Retype new UNIX password:
passwd: пароль успешно обновлён
```

Для изменения параметров учётной записи можно отредактировать файл /etc/passwd, однако более корректным способом

является использование команды usermod. Например, для изменения командной оболочки пользователя на /bin/false приведёт к невозможности его входа в систему: **usermod -s /bin/false testuser**.

Удаление пользователя производится с помощью команды userdel: **userdel testuser**.

Задания для самостоятельной работы

1. Выясните, чем отличается реакция операционной системы на различные ошибки аутентификации.
2. Сравните права доступа к директориям /bin и /tmp. Какие операции сможет совершать в них простой пользователь?
3. Создайте текстовый файл и задайте права на него таким образом, чтобы он мог просматриваться только владельцем и никем не мог редактироваться.
4. Что смогут делать другие пользователями с файлами в домашней директории пользователя, если он задаст всем остальным пользователям право на запись в директорию, но удалит право исполнения на неё?
5. Найдите все исполняемые файлы с установленным suid-битом.
6. Получите имена всех пользователей системы, у которых в качестве командной оболочки используется программа /bin/false.

Контрольные вопросы

1. Что необходимо ввести для входа в систему?
2. Для чего нужна команда rm?
3. С помощью какой команды производится изменение прав доступа?
4. Как перейти в режим суперпользователя?
5. Какая команда используется для добавления новых пользователей?
6. Какая команда используется для удаления пользователей?

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 10. «МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО КОМАНДАМ УПРАВЛЕНИЯ СЕТЬЮ В UNIX»

Теоретическая часть

Команды по конфигурированию сети

Для настройки сетевых интерфейсов используется команда ifconfig, имеющая следующий синтаксис:

ifconfig [-L] [-m] interface [create] [address_family] [address [dest_address]] [parameters] ifconfig interface destroy ifconfig -a [-L] [-d] [-m] [-u] [address_family] ifconfig -l [-d] [-u] [address_family] ifconfig [-L] [-d] [-m] [-u] [-C]

Команда может использоваться при загрузке системы для настройки адресов каждого сетевого интерфейса, а также после загрузки для изменения параметров сетевых интерфейсов. Если команда введена без аргументов, **ifconfig** выдает информацию о состоянии активных интерфейсов. Если в качестве аргумента указан какой-либо интерфейс, то выдается информация только о состоянии этого интерфейса; если указан один аргумент -a, выдается информация о состоянии всех интерфесов, даже отключенных. Пример:

```
user@desktop$ ifconfig rl0
rl0:
flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST>  mtu
1500
options=8<VLAN_MTU>
inet6 fe80::250:22ff:febb:5f1%rl0  prefixlen  64 scopeid  0x3
inet  192.168.19.86  netmask  0xffffff00  broadcast
192.168.19.255
ether 00:50:22:bb:05:f1
media: Ethernet autoselect (100baseTX <full-duplex>)
status: active
```

Иначе команда конфигурирует указанный интерфейс. Изменить настройки какого-либо интерфейса может только суперпользователь.

Опции:

интерфейс	– имя интерфейса (например, rl0 в BSD или eth0 в Linux).
up	– вызывает активизацию интерфейса. Задается неявно при присвоении адреса интерфейсу.
down	– вызывает остановку работы драйвера для интерфейса.
[-]arp	– включает или отключает использование протокола ARP для интерфейса.
[-]promisc	– включает или отключает неразборчивый режим (promiscuous mode) работы интерфейса. В этом режиме все проходящие по сети пакеты будут приниматься интерфейсом.
[-]allmulti	– включает или отключает режим all-multicast. В этом режиме все многоадресные (multicast) пакеты в сети будут приниматься интерфейсом.
metric N	– устанавливает метрику интерфейса.
mtu N	– устанавливает максимальный размер пакета (Maximum Transfer Unit - MTU) для интерфейса.
адрес	– IP-адрес, присваиваемый интерфейсу.
netmask адрес	– устанавливает маску сети IP для этого интерфейса. По умолчанию используется обычная маска сети класса A, B или C (что определяется по IP-адресу интерфейса), но можно усановить любое значение.
add адрес/длина_префикса	– добавляет адрес IPv6 для интерфейса.

del адрес/длина_префикса	– удаляет адрес IPv6 для интерфейса.
irq адрес	– устанавливает аппаратное прерывание, используемое устройством. Не для всех устройств можно динамически менять значение IRQ.
media тип	– устанавливает физический порт или тип носителя, используемый устройством. Не для всех устройств можно менять этот параметр, и для разных устройств могут поддерживаться различные значения. Типичные значения типа - 10base2 (коаксиальный кабель Ethernet), 10baseT (витая пара Ethernet 10 Мбит/сек), AUI (внешний передатчик) и т.д. Специальный тип носителя auto можно использовать, чтобы потребовать от драйвера автоматически обпределять тип носителя. Не все драйверы могут это делать.
[-]broadcast [адрес]	– если указан аргумент адрес, задает соответствующий протоколу широковещательный адрес для интерфейса. В противном случае устанавливает (или сбрасывает) флаг IFF_BROADCAST для интерфейса.

Пример изменения IP-адреса интерфейса rl0:

```
user@desktop ~ $ ifconfig rl0
rl0:
flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST>  mtu
1500
        options=8<VLAN_MTU>
        inet6 fe80::250:22ff:febb:5f1%rl0 prefixlen 64 scopeid
0x3
        inet 192.168.19.86 netmask 0xffffff00 broadcast
192.168.19.255
        ether 00:50:22:bb:05:f1
```

```
        media: Ethernet autoselect (100baseTX <full-duplex>)
        status: active
user@desktop ~ $ ifconfig rl0 192.168.0.1
user@desktop ~ $ ifconfig rl0
rl0:
flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST>  mtu
1500
        options=8<VLAN_MTU>
        inet6 fe80::250:22ff:febb:5f1%rl0 prefixlen 64 scopeid
0x3
        inet 192.168.0.1 netmask 0xffffff00 broadcast
192.168.19.255
        ether 00:50:22:bb:05:f1
        media: Ethernet autoselect (100baseTX <full-duplex>)
        status: active
```

Команда **arp** отображает ARP-таблицу данного хоста. с помощью параметра *-i* можно специфицировать сетевой интерфейс, информация о котором интересует.

```
desktop ~ # arp -i eth0
Address                  HWtype  HWaddress           Flags Mask
Iface
DIMON.mshome.net         ether   00:50:BF:12:8A:9E   C
eth0
```

Таблица с информацией о канальном уровне содержит связь IP- и MAC-адресов. При использовании параметра *-n* IP-адреса не будут заменяться символьными именами хостов.

Команда **route** используется для просмотра и изменения таблицы маршрутизации хоста. Для этой команды также работает параметр *-n*, при использовании которого IP-адреса не будут заменяться символьными именами хостов.

Пример обычной таблицы маршрутизации для отдельного компьютера в сети:

```
desktop ~ # route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref
Use Iface
192.168.5.0     0.0.0.0         255.255.255.0   U     0      0
0 eth1
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0
0 lo
0.0.0.0         192.168.5.254   0.0.0.0         UG    0      0
0
```

eth1

Особый интерес представляет адрес 0.0.0.0, который соответствует хосту назначения по умолчанию.

Для добавление нового маршрута к определённому хосту используются параметры *add* и *-host*:

desktop ~ # route add -host 192.168.0.1 eth0

Эта команда создаёт новую строку в таблице маршрутизации, согласно которой все пакеты к узлу 192.168.0.1 должны отправляться в сетевой интерфейс eth0.

Также можно добавлять шлюз для отправки пакетов в определённую сеть или к хосту:

desktop ~ # route add -net 192.168.1.0 gw 192.168.0.5

Таким образом, все пакеты для сети 192.168.1.0 будут направляться на узел 192.168.0.5.

Аналогично, маршруты удаляются параметром *del* с указанием всей информации о маршруте:

desktop ~ # route del default gw 192.168.0.1

Эта команда удаляет маршрут по умолчанию через хост 192.168.0.1.

Команды по диагностике сети

Для посылки пакетов ICMP ECHO_REQUEST сетевым хостам используется команда **ping,** имеющая следующий синтаксис:

ping [-AaDdfnoQqRrv] [-c число_пакетов] [-i секунд] [-l preload] [-M mask | time] [-m ttl] [-P policy] [-p pattern] [-S src_addr] [-s packetsize] [-t timeout] [-z tos] host ping [-AaDdfLnoQqRrv] [-c число_пакетов] [-I iface] [-i секунд] [-l preload] [-M mask | time] [-m ttl] [-P policy] [-p pattern] [-S src_addr] [-s packetsize] [-T ttl] [-t timeout] [-z tos] mcast-group

Команда **ping** использует датаграмму ECHO_REQUEST протокола ICMP, чтобы вызвать ответ ICMP ECHO_RESPONSE указанного хоста или сетевого шлюза. Если хост отвечает, **ping** выдает сообщение, что хост включен (хост is alive), в стандартный выходной поток.

Для проверки наличия хоста в сети достаточно ввести команду **ping** с аргументом - именем или адресом хоста:

user@desktop$ ping yandex.ru

64 bytes from 213.180.204.11: icmp_seq=0 ttl=48 time=5.659 ms

64 bytes from 213.180.204.11: icmp_seq=1 ttl=48 time=5.404 ms

64 bytes from 213.180.204.11: icmp_seq=2 ttl=48 time=4.889

ms

^C

--- yandex.ru ping statistics ---

3 packets transmitted, 3 packets received, 0% packet loss

round-trip min/avg/max/stddev = 4.889/5.317/5.659/0.320 ms

Для отправки определенного числа пакетов необходимо указать опцию -c <число_пакетов>. Для установки интервала между отправкой пакетов используется опция -i <количество секунд>.

Для отладки сетевых соединений посредством построения маршрута следования пакетов к хосту назначения служит команда **traceroute**. Для этой команды также работает параметр -n, при использовании которого IP-адреса не будут заменяться символьными именами хостов.

Пример следования пакетов до хоста ya.ru:

desktop ~ # traceroute ya.ru

traceroute to ya.ru (213.180.204.8), 64 hops max, 40 byte packets

1 195.91.230.65 (195.91.230.65) 0.890 ms 1.907 ms 0.809 ms

2 cs7206.rinet.ru (195.54.192.28) 0.895 ms 0.769 ms 0.605 ms

3 ix2-m9.yandex.net (193.232.244.93) 1.855 ms 1.519 ms 2.95 ms

4 c3-vlan4.yandex.net (213.180.210.146) 3.412 ms 2.698 ms 2.654 ms

5 ya.ru (213.180.204.8) 2.336 ms 2.612 ms 3.482 ms.

Команда **netstat** используется для показа состояния сети и имеет следующий синтаксис:

netstat [-AaLnSW] [-f protocol_family | -p protocol] [-M core] [-N system]

Команда **netstat** показывает содержимое различных структур данных, связанных с сетью, в различных форматах в зависимости от указанных опций. Первая форма команды показывает список активных сокетов (sockets) для каждого протокола. Вторая форма выбирает одну из нескольких других сетевых структур данных. Третья форма показывает динамическую статистику пересылки пакетов по сконфигурированным сетевым интерфейсам; аргумент интервал задает, сколько секунд собирается информация между последовательными показами.

Опции:

-a	– показывать состояние всех сокетов; обычно сокеты, используемые серверными процессами, не показываются.
-A	– показывать адреса любых управляющих блоков протокола, связанных с сокетами; используется для отладки.
-i	– показывать состояние автоматически сконфигурированных (auto-configured) интерфейсов. Интерфейсы, статически сконфигурированные в системе, но не найденные во время загрузки, не показываются.
-n	– показывать сетевые адреса как числа. netstat обычно показывает адреса как символы. Эту опцию можно использовать с любым форматом показа.
-r	– показать таблицы маршрутизации. При использовании с опцией -s, показывает статистику маршрутизации.
-s	– показать статистическую информацию по протоколам. При использовании с опцией -r, показывает статистику маршрутизации.
-f семейство_адресов	– ограничить показ статистики или адресов управляющих блоков только указанным семейством_адресов, в качестве которого можно указывать: net Для семейства адресов AF_INET nix Для семейства адресов AF_UNIX

-I интерфейс	– выделить информацию об указанном интерфейсе в отдельный столбец; по умолчанию (для третьей формы команды) используется интерфейс с наибольшим объемом переданной информации с момента последней перезагрузки системы. В качестве интерфейса можно указывать любой из интерфейсов, перечисленных в файле конфигурации системы, например, emd1 или lo0.
-p имя_протокола	– Ограничить показ статистики или адресов управляющих блоков только протоколом с указанным именем_протокола, например, tcp.

Пример показа таблицы маршрутизации:

```
user@desktop ~$ netstat -r
Routing tables
Internet:
Destination        Gateway            Flags     Refs      Use   Netif
Expire
default            19-101.local       UGS          0  1373769     rl0
localhost          localhost          UH           1      290     lo0
192.168.0          link#1             UC           0        0     dc0
192.168.19         link#3             UC           0        0     rl0
19-86.local        localhost          UGHS         0        0     lo0
19-101.local       00:0d:bc:e4:27:bf  UHLW         1        0     rl0
116

Internet6:
Destination        Gateway            Flags      Netif  Expire
localhost.prov.ru  localhost.prov.ru  UH           lo0
fe80::%dc0         link#1             UC           dc0
fe80::2a0:ccff:fe3 00:a0:cc:3d:1f:bd  UHL          lo0
fe80::%rl0         link#3             UC           rl0
fe80::250:22ff:feb 00:50:22:bb:05:f1  UHL          lo0
fe80::%lo0         fe80::1%lo0        U            lo0
fe80::1%lo0        link#5             UHL          lo0
ff01::             localhost.prov.ru  U            lo0
ff02::%dc0         link#1             UC           dc0
```

```
ff02::%rl0        link#3            UC          rl0
ff02::%lo0        localhost.prov.ru UC          lo0
```

Команда **host** служит для получения доменной информации о хосте: IP-адреса, MX-записи и другой информации, связанной с данным символьным именем. Имя хоста указывается в качестве аргумента команды.

Пример работы команды:

```
user@desktop ~$ host yandex.ru
yandex.ru has address 213.180.204.11
yandex.ru mail is handled by 10 mx2.yandex.ru.
yandex.ru mail is handled by 0 mx1.yandex.ru.
```

Вторым аргументом можно указать DNS-сервер, который будет использоваться при получении этой информации:

```
user@desktop ~$ host yandex.ru ns1.aiya.ru
Using domain server:
Name: ns1.aiya.ru
Address: 85.142.20.152#53
Aliases:

yandex.ru has address 213.180.204.11
Using domain server:
Name: ns1.aiya.ru
Address: 85.142.20.152#53
Aliases:

Using domain server:
Name: ns1.aiya.ru
Address: 85.142.20.152#53
Aliases:

yandex.ru mail is handled by 0 mx1.yandex.ru.
yandex.ru mail is handled by 10 mx2.yandex.ru.
```

Команда **tcpdump** используется для мониторинга сети на канальном и более высоких уровнях. Программа «слушает» на одним или нескольких сетевых интерфейсах и выводит дамп пакетов, проходящих через этот интерфейс.

Параметр *-i* задаёт имя сетевого интерфейса, на котором запускается прослушивание. При просмотре захватываемых данных удобно использовать ключ *-l*, который буферизует вывод построчно. Для этой команды также работает параметр *-n*, при ис-

пользовании которого IP-адреса не будут заменяться символьными именами хостов.

Пример работы команды:

desktop ~ # tcpdump -i eth0 -l -n

tcpdump: verbose output suppressed, use -v or -vv for full protocol decode

listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes

12:51:07.486755 arp who-has 0.0.0.0 (00:30:48:2b:6d:6a) tell 0.0.0.0

12:51:12.486606 arp who-has 0.0.0.0 (00:30:48:2b:6d:6a) tell 0.0.0.0

12:51:14.457608 IP 192.168.5.23.56385 > 194.91.250.11.443: P 3645922938:3645923156(218) ack 2092518729 win 10086

12:51:14.491343 IP 194.91.250.11.443 > 192.168.5.23.56385: . ack 218 win 10720

Для вывода расширенной информации о пакетах используются ключи *-v* или *-vv*.

desktop ~ # tcpdump -i eth1 -l -n -vv

tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes

12:57:53.043797 IP (tos 0x0, ttl 51, id 46031, offset 0, flags [DF], proto: TCP (6),

length: 286) 194.91.250.11.5190 > 192.168.5.23.38993: P 2517343058:2517343292(234)

ack 2346573376 win 2202 <nop,nop,timestamp 2713588760 497668>

12:57:53.043865 IP (tos 0x0, ttl 64, id 52382, offset 0, flags [DF], proto: TCP (6),

length: 52) 192.168.5.23.38993 > 194.91.250.11.5190: ., cksum 0x1fd7 (correct),

1:1(0) ack 234 win 11945 <nop,nop,timestamp 506366 2713588760>

12:57:53.401516 IP (tos 0x0, ttl 48, id 45237, offset 0, flags [DF], proto: TCP (6),

length: 210) 194.91.250.11.443 > 192.168.5.23.56385: P 2092522043:2092522213(170)

ack 3645927446 win 10720

...

Команда **tcpdump** обладает очень богатым интерфейсом, включающим условные выражения, по которым должны выделяться интересующие пакеты. Например, можно использовать

условия удалённого порта (равно 80):

desktop ~ # tcpdump -i eth1 -l -n -vv dst port 80

tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes

13:55:36.563959 IP (tos 0x0, ttl 64, id 3936, offset 0, flags [DF], proto: TCP (6),

length: 60) 192.168.5.23.52348 > 213.180.204.11.80: S, cksum 0x2766 (correct),

3855548287:3855548287(0) win 5840 <mss 1460,sackOK,timestamp 1372191 0,nop,wscale 2>

13:55:36.592654 IP (tos 0x0, ttl 64, id 3937, offset 0, flags [DF], proto: TCP (6),

length: 40) 192.168.5.23.52348 > 213.180.204.11.80: ., cksum 0xebc5 (correct),

3855548288:3855548288(0) ack 3869420799 win 5840

13:55:36.592731 IP (tos 0x0, ttl 64, id 3938, offset 0, flags [DF], proto: TCP (6),

length: 627) 192.168.5.23.52348 > 213.180.204.11.80: P 0:587(587) ack 1 win 5840

Команда **nmap** – сетевой сканер, с помощью которого можно определить уязвимость удалённых хостов. Основное назначение этой программы – определение состояние портов удалённого хоста (закрыты они, открыты или заблокированны). Также программа может на основании собственной базы знаний определить по поведению удалённого хоста, какая операционная система запущена на нём.

Команды удалённого терминала

Программа ssh является более современным и защищённым аналогом программы telnet.

Команды по управлению сетевым экраном

Команда **iptables** является интерфейсом к межсетевому экрану ядра Linux.

Контрольные вопросы

1. Какая команда используется для настройки сетевых интерфейсов?
2. Для чего используется команда route?

3. Расскажите о командах диагностики сети.
4. Какие существуют команды удалённого терминала?
5. Расскажите о команде iptables.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 11. «РАБОТА В КОМАНДНОЙ СТРОКЕ WINDOWS»

Теоретическая часть

Командная строка является программной оболочкой позволяющей в текстовом виде вводить компьютеру различные команды. Когда привычного всем графического интерфейса в операционных системах не было, все делалось в командной строке, и именно благодаря появлению графического интерфейса в виде привычных нам сейчас окон, Windows завоевала огромную популярность во всем мире.

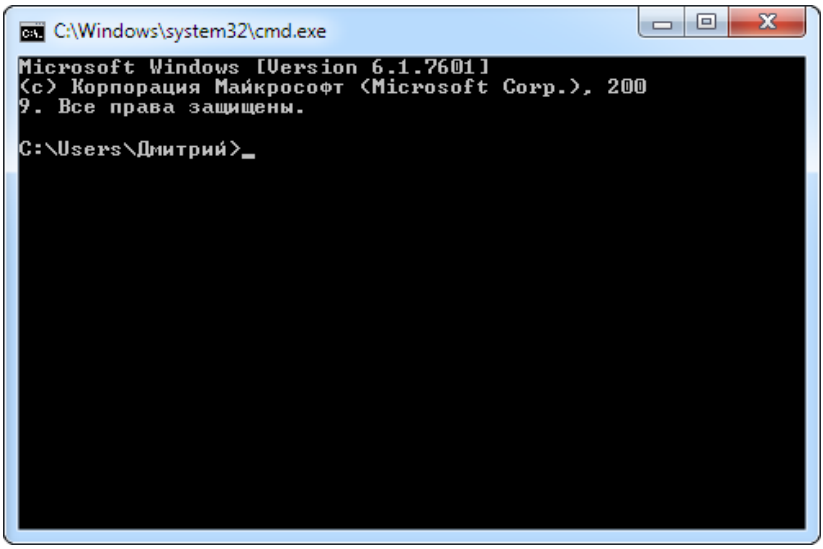
Даже в настоящее время некоторые команды быстрее выполнить в командной строке, а некоторые настройки в принципе отсутствуют в графическом интерфейсе пользователя. Так же следует иметь в виду, что до сих пор существуют утилиты, не имеющие графического интерфейса, а иногда он оказывается недоступен, например из-за сбоя.

Как запустить командную строку

Способов запустить командную строку несколько, причем они могут несколько различаться в разных версиях Windows. Перечислим несколько способов:

1. Нажмите сочетание клавиш WIN+R, введите cmd и нажмите кнопку Ok;
2. Нажмите кнопку «Пуск» введите в поле поиска «командная строка» и щелкните в результатах поиска «Командная строка» (можно ввести «cmd» и выбрать в результатах «cmd»);
3. «Пуск» ⇒ «Все программы» ⇒ «Стандартные» ⇒ «Командная строка»;
4. Запустить файл C:\Windows\System32\cmd.exe.

В независимости от использованного способа у вас откроется окно командной строки.



Теперь можно управлять системой, набирая в ней нужные команды и смотреть на результат. Итог выполнения команды будет зависеть от самой команды.

Например, выполнение команды notepad запустит Блокнот.

Таким образом можно запустить любую программу или исполняемый файл, но в большинстве случаев потребуется ввести полный путь. Если в пути есть пробелы, весь путь необходимо заключить в кавычки. Например:

"C:\Program Files (x86)\Mozilla Firefox\firefox.exe"

Так же можно выполнять различные операции с папками и файлами, такие как создание, удаление, копирование, переименование и так далее.

Если команда введена неверно или ее выполнение невозможно по каким-либо причинам, то в командной строке появится сообщение об ошибке.

Так же некоторые команды могут выполняться совсем без внешних видимых изменений в системе или самой командной строке. Другие наоборот, требуют реакции пользователя в процессе выполнения, выводя соответствующие запросы.

Наиболее часто использующиеся команды:

cd - для смены текущего каталога;

cd /? - краткая справка по команде cd;

chdir или cd /d i:\Games – смена текущего диска;

dir – отображение содержимого текущего каталога;

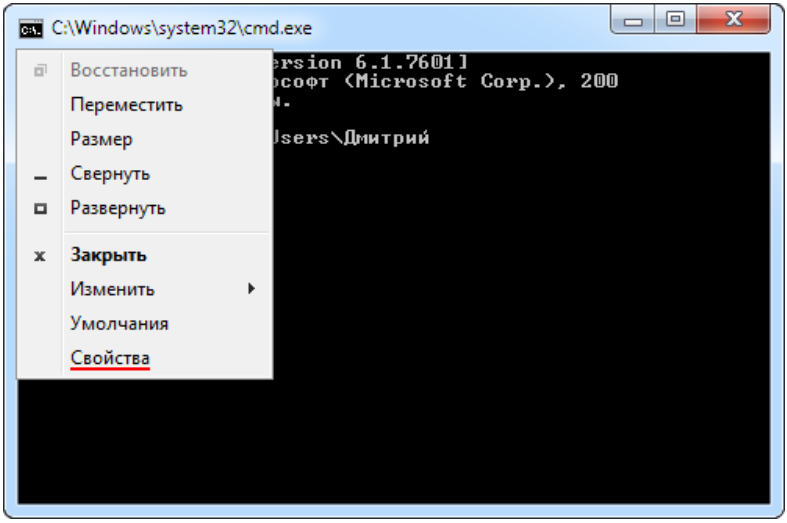
md – создание папки;
move – переименование или перемещение файла;
rd – удаление каталога.

Альтернативный способ заключается в том, чтобы сразу запустить командную строку Windows в нужном месте. Для этого необходимо открыть нужную папку в Проводнике, щелкнуть на свободном месте правой кнопкой мыши с нажатой клавишей Shift и выбрать в контекстном меню «Открыть окно команд».

Стоит отметить, что команды не обязательно вводить непосредственно в командной строке Windows, вместо этого их можно записать в так называемый командный файл с расширением .bat или .cmd и запускать его, когда потребуется.

Настройка командной строки Windows

Для настройки командной строки щелкните по значку командной строки в левом верхнем углу окна и выберите пункт «Свойства».



В открывшемся окне можно подобрать наиболее подходящие вам параметры для комфортной работы, например, размеры окна или цвет шрифта.

Если в режиме ожидания ввода команды нажимать клавишу ↑ на клавиатуре, то будут подставляться введенные вами ранее

команды. Настроить данный параметр можно на вкладке «Общие» в блоке «Запоминание команд».

В соседнем блоке «Правка» можно отметить чекбоксы «Выделение мышью» и «Быстрая вставка», которые позволяют выполнять копирование и вставку с помощью правой клавиши мыши.

Запуск командной строки с правами администратора

Для выполнения некоторых команд требуются права администратора. Причем просто запустить командную строку Windows находясь в учетной записи пользователя с административными правами недостаточно. Ее необходимо запустить от имени администратора. Сделать это можно следующими способами:

1. Нажмите кнопку «Пуск» введите в поле поиска «командная строка» и щелкните правой кнопкой мыши по строчке «Командная строка» в результатах поиска и выберите в контекстном меню «Запуск от имени администратора» (все то же самое можно сделать с «cmd»);
2. «Пуск» ⇒ «Все программы» ⇒ «Стандартные» ⇒ щелкнуть правой кнопкой мыши по «Командная строка» и выбрать «Запуск от имени администратора»;
3. Открыть в Проводнике папку C:\Windows\System32, щелкнуть по файлу cmd.exe правой кнопкой мыши и выбрать пункт «Запуск от имени администратора»;
4. Создать ярлык для запуска командной строки, щелкнуть по нему правой клавишей мыши и выбрать «Запуск от имени администратора»;
5. Создать ярлык для командной строки и в его свойствах указать «Запускать от имени администратора», теперь командная строка будет сразу запускаться с нужными правами;
6. Нажать комбинацию клавиш WIN+X (в Windows 8).

Порядок выполнения работы

1. Запустите командную строку.

Задание №1.

Выйдите в корневой каталог диска **C:**. Зайдите на этом диске в папку **Program Files**. Выйдите из каталога**Program Files.** Перейдите на диск **D:**.

Если необходимо очистить экран от старых записей, воспользуйтесь командой **CLS.**

Создание папки (каталога). Переименование.

На диске D необходимо создать папку **Practica**.

Команда указывается без имени диска, если мы находимся на этом диске. Если же мы на другом диске, например на диске C, то формат команды будет такой: **Md_D:\Practica.**

В каталоге **Practica** Создайте каталог **Ученик**.

Каталог **Ученик** Переименовать в каталог **Студент**.

Зайдите на диск D, просмотрите полученный результат.

Создайте структуру каталогов D:\Иванов\Иван\Иванович

Задание №2.

Создайте структуру каталогов: C:\task\temp\temp1. Переименуйте каталог temp1 в builder. Зайдите на диск C, просмотрите полученный результат.

Удаление каталогов.

При удалении каталога необходимо придерживаться следующего правила: каталог должен быть пуст и не должен являться текущим.

Удалите структуру каталогов: C:\task\temp\temp1.

Задание №3.

Удалите структуру каталогов **D:\Practica\Студент.** Удалите структуру каталогов **C:\Task\Temp\Builder**

Работа с файлами. Создание файла.

На диске D создайте файл **Практика.Txt,** Содержащий текст **«Задание выполнено».**

Для этого набираем команду **Copy Con Практика.Txt.**

После выполнения команды вводим текст, который будет в этом файле: «Задание выполнено» и для сохранения файла нажимаем комбинацию CTRL+Z.

Для просмотра содержимого файла используется команда **Type путевое имя файла.**

Чтобы просмотреть содержимое файла Практика. txt наберите команду

Type Практика.Txt

Задание №4.

На диске C создайте файл **100.Txt**, содержащий текст «Интерфейс командной строки». Просмотрите содержимое этого фай-

ла.

Переименование файла.

Переименовать файл **Практика.Txt** в **Example.Txt**.

Для этого набираем команду **Ren Практика.Txt Example.Txt**.

Кроме смены имени можно изменить и расширение: файл Example.txt переименовать в Example.doc.

Для этого набираем команду **Ren Example.Txt Example.Doc.**

Удаление файла

Удалить файл **Example.Doc.**

Для этого набираем команду **Del Example.Doc.**

Объединение файлов. Копирование файлов.

Создайте на диске D следующие файлы:

1.Txt, содержащий текст «Работа с»,

2.txt, содержащий текст «командной»,

3.Txt, содержащий текст «строкой».

Необходимо объединить все эти файлы в файл **4.Txt** (он может быть как существующим, так и нет).

Для этого набираем команду: **Copy 1.Txt + 2.Txt + 3.Txt 4.Txt**

Просмотрите содержимое полученного файла: в нем должен быть текст со всех объединенных файлов, т.е. в нем должна быть строка «Работа с командной строкой».

Скопировать полученный файл **4.Txt** на диск **C**.

Для этого набираем команду **Copy 4.Txt C:**

Скопировать все файлы с расширением **Txt** из папки **Windows** в папку **COPYR** (предварительно ее создав).

Затем набираем следующую команду: **Copy C:\Windows*.txt COPYR.**

Задание №5.

Создать файл с путевым именем: D:\100\200\300.Txt, затем все удалить по правилу удаления.

1. Вначале перейдем на диск D.
2. Затем создаем структуру каталогов: md_100/200
3. Теперь создадим файл, для этого укажем его полное путевое имя: copy_con_100\200\300.txt.
4. Для его сохранения нажимаем CTRL+Z.

Зайдите на диск D и проверьте результат.

Удаление:

1. Вначале удалим файл: del_100\200\300.txt
2. Затем удалим папку 200: rd_100\200
3. Теперь удалим папку 100: rd_100.

Зайдите на диск D и проверьте результат.

Итоговое задание:

Создать файл с путевым именем **D:\Tom1\Tom2\Test.Txt**, содержащий текст «**Итоговое задание**». Переименовать файл **Test.Txt** в **Samba.Doc** . Просмотреть содержимое этого файла. Показать преподавателю. Удалить все по правилу удаления. Создать на диске **D** каталог **Master**. Скопировать в него все файлы с расширением **Gif** из папки **C:\Windows**.

Контрольные вопросы:

1. Какие способы для запуска командной строки вы знаете?
2. Какая команда позволяет сразу выйти в корневой каталог?
3. Назовите команду для создания каталога?
4. Какое вы знаете правило для удаления каталогов?
5. Какая команда создает файл?
6. Какая команда удаляет файл?

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 12. «ПРОГРАММИРОВАНИЕ КОМАНДНЫХ BAT-ФАЙЛОВ В WINDOWS»

Теоретическая часть

Командный интерпретатор command.com

В операционных системах MS-DOS и Windows команды выполняются с помощью специальной программы – интерпретатора команд *command.com*. Некоторые команды распознаются и выполняются самим командным процессором – они называются *внутренними*, другие представляют собой отдельные программы – их называют *внешними (утилитами)*.

Внутренними командами являются:

Команда	Описание
BREAK	прерывание
CALL	вызов командного файла
CHCP	вывод или смена текущей кодовой страницы
CHDIR	смена каталога
CLS	очистка экрана
COPY	копирование файлов
CTTY	смена устройства вывода – экран, принтер и т.д.
DATE	просмотр и установка даты
DEL	удаление файла
DIR	вывод информации о содержимом дисков и каталогов
ECHO	вывод строки на экран
EXIT	выхода из интерпретатора command.com
FOR	оператор цикла в bat-файлах
GOTO	оператор перехода в bat-файлах
IF	условный оператор в bat-файлах
LOADHIGH	загрузка программы в область верхней памяти
MKDIR	создание каталога
PATH	указание списка путей для автоматического поиска

PAUSE	пауза до нажатия любой клавиши
PROMPT	установка формата командной строки
REM	комментарий в командных файлах
RENAME	переименование файла
RMDIR	удаления каталога
SET	установка переменных окружения
SHIFT	сдвиг параметров для пакетного файла
TIME	просмотр и установка времени
TYPE	просмотр файла
VER	вывод версии
VERIFY	контроль записи
VOL	информация о томе

Все остальные команды являются внешними и хранятся в виде исполняемых файлов в папке *C:|Windows|Command*.

Для работы с командной строкой в Windows можно выполнить *Пуск/Программы/Сеанс Ms-Dos,* запустить файл *command.com* непосредственно или использовать файловые менеджеры FAR, Norton Commander. При запуске *command.com* можно использовать ключи (кстати, их мы уже упоминали в параметре SHELL файла config.sys). Кроме указанных там параметров /P (загрузка в память) и /E (размер области переменных окружения) интересны параметры /Y (выполнение пакетного файла по шагам – для отладки), /C (исполнение указанной далее программы и возврат), /K (исполнение команды с продолжением). Используя ключ /C, например, можно создавать ярлыки для выполнения внутренних команд.

Отметим сначала несколько особенностей опередения путей к файлам Windows. Файловая система имеет древовидную структуру и имена файлов задаются в формате *[диск:] [путь|]имя_файла*. Если путь начинается с символа «\», то маршрут вычисляется от корневого каталога, иначе от текущего.

Существуют особые обозначения для текущего каталога («.») и трех его верхних уровней («..» — родительский, «…» — второго уровня и «….» — третьего уровня). Например, для текущего каталога C:\Windows\Media\Office97 путь к файлу autoexec.bat в корневом каталоге диска C: может быть записан в виде

….\autoexec.bat.

В именах файлов можно применять шаблоны * (произвольное кол-во любых символов) или ? (один произвольный символ или его отсутствие). Для того чтобы использовать длинные имена файлов при работе с командной строкой, их нужно заключать в двойные кавычки.

В синтаксисе командной строки особое место занимает перенаправление ввода-вывода с помощью символа «>» на стандартное устройство (PRN, COM1 и т. п) или файл. Например, **DIR /? > helpdir.txt** выведет справку по команде DIR в файл. Символ «>>» позволяет не создавать файл заново, а дописать в него. По аналогии символ «<» позволяет читать данные не с клавиатуры, а с файла. Например, **DATE < date.txt** ввод новой даты из файла.

Кроме того, командная строка поддерживает конвейеризацию с помощью конструкции: команда1 | команда2, когда сообщения выводимые первой командой используются в качестве входных данных для второй. Например, команда MORE выводит информацию частями, не превышающим размер экрана. Строка **TYPE имя_ файла | MORE** выводит файл по экранам с паузой.

Переменные окружения

При загрузке ОС Windows в оперативной памяти постоянно хранится набор т.н. *переменных окружения* (environment variables). Хотя в Windows есть более совершенный способ для хранения системных значений – реестр, многие программы по-прежнему используют переменные окружения.

Наиболее важные переменные хранят системный путь для поиска (PATH), каталог запуска Windows (WINDIR), место хранения временных файлов (TEMP).

Переменные устанавливаются с помощью команды

SET [переменная[строка]]

Запуск SET без параметров приводит к выводу списка переменных среды. Для получения их значений (всегда строки) нужно имя соответствующей переменной заключить в символы «%», например: %TEMP%.

Основные команды

Остановимся подробнее на командах работы с файловой системой: ATTRIB, COPY, XCOPY, DIR, MKDIR, RMDIR, DEL,

DELTREE, REN, MOVE, SUBST, VOL, LABEL.

ATTRIB [+R | -R] [+A | -A] [+S | -S] [+H | -H] [[диск:][путь]имя_файла] [/S] - просмотра или установка атрибутов файлов.

В Windows 9x можно использовать 4 атрибута:

- Read-Only (R) – только для чтения;
- System (S) – системный;
- Archive (A) – архивный;
- Hidden (H) – скрытый.

Установка атрибута производится ключом «+», снятие «-». Ключ «/S» применяется для обработки файлов во всех подкаталогах указанного пути. Например, сделать все Word-файлы в каталоге «Мои документы» доступными только для чтения: **ATTRIB +R "C:\Мои документы *.doc"**. Атрибуты фалов можно просмотреть/поменять в файловых менеджерах или проводнике (пункт «Свойства» в контекстном меню по правой клавише).

COPY [/A | /B] источник [/A | /B] [+источник [/A | /B] [+ …]] [результат [/A | /B]] [/V] [/Y | /-Y] – копирование файлов.

Описание ключей сведено в таблицу:

Параметр	**Описание**
/A	файл является текстовым файлом ASCII (символ + в конце)
/B	файл является двоичным, результат - каталог для результата копирования и/или имя создаваемого файла
/V	проверка правильности копирования путем сравнения источника и приемника
/Y	отключение режима запроса подтверждения на замену файлов
/-Y	включение режима запроса подтверждения на замену файлов

В качестве источника или результата при копировании можно указывать имена не только файлов, но и устройств компьютера:

- PRN – принтер,
- LPT1-LPT3 – соответствующие параллельные порты;
- AUX – устройство, присоединяемое к последовательному

порту 1;

• COM1-COM3 — соответствующие последовательные порты;

• CON – терминал (при вводе – это клавиатура, а при выводе – монитор);

• NUL – пустое устройство (все операции игнорируются).

Например, для печати файла на принтере можно ввести команду **COPY abc.txt PRN**, для создания файла **COPY CON my.txt** (для конца файла ввести +).

Команда COPY может объединять (склеивать) несколько файлов путем использования групповых знаков «*» и «?» (**COPY /B *.dat all.dt**) или знака «+» (**COPY /B 1.txt+2.txt 3.txt**). Здесь ключ /B используется для предотвращения усечения соединяемых файлов, т.к. по умолчанию файлы считаются текстовыми.

Среди недостатков команды COPY надо отметить невозможность копирования скрытых системных файлов, замены файлов с атрибутом «Read Only», а также копирования открытых (занятых) файлов (в этом случае процесс просто прерывается).

Некоторые из этих проблем можно решить с помощью утилиты XCOPY.

XCOPY источник результат [/A | /M] [/D[:дата]] [/P] [/S [/E]] [/W] [/C] [/I] [/Q] [/F] [/L] [/H] [/R] [/T] [/U] [/K] [/Y | /-Y]

Команда XCOPY работает только с файлами и каталогами (а не с устройствами).

Используемые ключи рассмотрим в табличном виде:

Ключ	Описание
/A	Копирование только файлов с установленным архивным атрибутом. Сам атрибут при этом не изменяется.
/M	Копирование только файлов с установленным архивным атрибутом. После копирования атрибут снимается.
/D	Копирует только файлы, измененные не ранее указанной даты. Если дата опущена – то если источник новее результата.
/S	Копирование каталогов с подкаталогами (кроме пустых).

/E	Копирование каталогов с подкаталогами, в том числе пустых
/W	Режим запроса на нажатие любой клавиши до начала копирования
/C	Продолжение в случае возникновения ошибок
/I	Копирование нескольких файлов, когда файл назначения отсутствует. В этом случае считает, что файл назначения д.б. каталогом и создает его без дополнительных запросов
/V	Сравнение конечных файлов с исходными
/P	Вывод запросов перед созданием каждого нового файла
/Q, /F, /L	Запрет вывода имен копируемых файлов, /L – вывод имен, /F – полных
/G (2000)	Копирование зашифрованных файлов в каталог результата, не поддерживающий шифрование
/H	Копирование скрытых и системных файлов (среди прочих)
/R	Разрешение замены файлов, предназначенных только для чтения
/T	Создание структуры каталогов (кроме пустых каталогов) без копирования файлов. Если требуется создать пустые каталоги и подкаталоги — /T /E.
/U	Копирование только файлов, уже имеющихся в конечном каталоге
/K	Копирование атрибута «Только чтение» (обычно он сбрасывается)
/N	Использование коротких имен при копировании

/O (NT)	Копирование сведений о владельце и данных ACL
/X (NT)	Копирование параметров аудита файлов (требует /O)
/Y /-Y	Подавление (обязательный) запроса подтверждения на перезапись существующего конечного файла
/Z (NT)	Копирование сетевых файлов с возобновлением
/EXCLUDE:файл1[+файл2]... (NT)	Исключение определенных файлов из операции копирования

При использовании команды XCOPY внутри пакетных файлов по переменной **ERRORLEVEL** можно анализировать код выхода (завершения) команды (0 – без ошибок, 1 – файлы не найдены, 2 – прерывание по +, 4 – не хватает места, ошибка в назначении или синтаксисе, 5 – ошибка записи на диск).

DIR [диск:][путь][имя_ файла] [/A[[:]атрибуты]] [/B] [/C] [/D] [/L] [/N] [/O[[:]порядок]] [/P] [/Q] [/S] [/T[[:]время]] [/W] [/X] [/4] - вывод информации о содержимом дисков и каталогов.

При использовании без ключей выводит метку диска, имена (в коротком и длинном вариантах) файлов и подкаталогов, находящихся в текущем подкаталоге, а также дату и время их последней модификации. Также выводятся общее число файлов в каталоге, их объем и размер свободного пространства.

Ключи команды DIR позволяют задать различные режимы расположения, фильтрации и сортировки выводимой информации:

Ключ	**Описание**
/A	Вывод файлов с указанными атрибутами (D-каталоги, A, R, S, H – как обычно. Префикс «-» будет иметь значение НЕ.
/B	Вывод только имен файлов (переопределяет /W)
/C	Применение разделителя групп разрядов для вывода размеров файлов (по умолчанию). Для отключения /-C.

/D (NT)	Вывод списка в несколько столбцов с сортировкой по столбцам
/L	Использование нижнего регистра для имен файлов
/N	Отображение имен файлов в крайнем правом столбце
/O	Сортировка списка отображаемых файлов (префикс «-» обращает порядок): N — по имени (алфавитная) S — по размеру (сперва меньшие) E — по расширению (алфавитная) D — по дате (сперва более старые) G — начать список с каталогов A – по дате загрузки (начиная с более старых)
/P	Пауза после заполнения каждого экрана
/Q (NT)	Вывод сведений о владельце файла
/V (9x)	Вывод расширенных сведений о файлах и каталогах
/S	Вывод списка файлов из указанного каталога и его подкаталогов
/T	Выбор поля времени для отображения и сортировки C – создание A — последнее использование W — последнее изменение
/W	Вывод списка в несколько столбцов
/X	Отображение коротких имен для файлов, чьи имена не соответствуют стандарту
/4 (9x)	Вывод номера года в четырехзначном формате

Стандартный набор ключей можно записать в переменную среды DIRCMD. Для отмены их действия введите в команде те же ключи с префиксом «-», например: /-W. Команда DIR (так же как и другие) поддерживает перенаправление ввода-вывода (символы

«>» и «>>»), например, команда

DIR "C:\Мои документы" /W/O:N > PRN

выполняет печать в широком формате в алфавитном порядке на принтер.

MKDIR (MD) [диск:]путь и **RMDIR (RD)** [диск:]путь - создание и удаление каталога. Команда MKDIR не будет выполнена, если каталог или файл с заданным именем уже существуют, а RMDIR – если удаляемый каталог не пустой. В Windows NT появилась возможность удалять подкаталоги (**RD /S**), и создавать полный путь – **MD \A\B\C\D**.

DEL [диск:][путь]имя_ файла [/P] и **ERASE** [диск:][путь]имя_ файла [/P] - удаление файлов. Ключ /P означает выдачу запроса на удаление (при задании удаления всех файлов **DEL *.*** или **DEL .** запрос будет всегда). В Windows NT добавлены параметры для удаления файлов с атрибутом «Только для чтения» (/F), подкаталогов (/S), отмены подтверждений (/Q), а также удаления файлов, не имеющих заданных атрибутов (/A:атрибуты).

DELTREE [/Y] [диск:]путь - удаление каталога вместе со всем содержимым. Ключ /Y используется для отключения запроса на подтверждение.

RENAME (REN) [диск:][путь][каталог1 |файл1] [каталог2 | файл2] - переименование файла или каталога. Так же можно использовать групповые символы «*» и «?», например: **REN *.txt *.doc**. В этой команде нельзя указать другой диск или каталог – для этих целей надо использовать команду MOVE.

MOVE [/Y | /-Y] [диск:][путь]имя_файла1[,...] рез_ файл - перемещение файлов или каталогов.

SUBST [диск1: [диск2:]путь] - сопоставление заданному пути имя виртуального диска. Часто нужно для инсталляции программ (эмуляции корневого каталога CD). Например: **SUBST F: C:\INSTALL\DELPHI5**. Ключ /D используется для удаления ранее созданного виртуального диска: **SUBST F: /D**. SUBST без параметров выводит текущий список виртуальных дисков (нельзя назначать их на сетевые каталоги).

VOL [диск:] и **LABEL** [диск:][метка] – вывод и задание метки тома.

Командные bat-файлы

Командный (пакетный) файл в Windows – это обычный текстовый файл с расширением *bat*, в котором записаны допустимые команды ОС, а также некоторые инструкции (ключевые слова) для алгоритмизации действий.

Например, deltmp.bat удалит все временные файлы в каталоге Windows\Temp:

```
C: |
CD %TEMP%
ATTRIB –R *.tmp
DEL *.TMP
```

Основной командой для вывода информации в пакетных файлах служит **ECHO [сообщение].**

По умолчанию команды пакетного файла перед исполнением выводятся на экран. С помощью команды **ECHO OFF** можно отключить дублирование вывода команд на экран, **ECHO ON** восстанавливает режим дублирования. Кроме этого, можно отключить дублирование отдельной строки, если предварить ее символом «@». И, наконец, может вывести пустую строку комбинацией **ECHO.** (с точкой в конце).

```
@ECHO OFF
ECHO Привет !
ECHO.
ECHO Пока …
```

При запуске командных файлов в командной строке можно указывать произвольное число параметров, которые затем можно использовать внутри пакетного файла.

Для доступа к параметрам применяются символы %0 – имя файла, %1-%9 – значения первых девяти параметров соответственно. Например, имеется командный файл copier.bat следующего содержания:

```
@ECHO oFF
CLS
ECHO Файл %0 копирует каталог %1 в %2
XCOPY %1 %2 /S
```

При запуске его из командной строки с 2 параметрами:

copier.bat C:|Programs D:|Backup

на экран выводится сообщение «Файл copier.bat копирует каталог C:\Programs в D:\Backup» и происходит соответствующее

копирование.

При необходимости можно использовать более девяти параметров командной строки с помощью команды **SHIFT**, которая изменяет значения замещаемых параметров с %0 по %9, копируя каждый параметр в предыдущий. Значение %1 – в %0, %2 в %1 и т.д., 10 параметр в %9. Команда, обратная SHIFT, отсутствует, поэтому восстановить параметры уже не удастся.

В командных файлах можно использовать переменные окружения и объявлять собственные с помощью команды SET. Все переменные рассматриваются как строки и в Windows 9x над ними нельзя производить арифметические действия (в отличие от NT), а только конкатенацию (просто слитно 2 переменные без знака «+»). Например, в результате выполнения командного файла:

SET A=Первый

SET B=Второй

SET C=%A%%B%

ECHO Переменная C=%C%

на экран выведется «Переменная C= Первый Второй».

Для управления выполнением командных файлов существуют команды приостановки, перехода, условного ветвления, циклов и вызова внешних командных файлов.

Для того, чтобы прервать выполнение командного файла, надо нажать + или + (в Windows NT есть команда выхода EXIT /B). Для приостановки с выдачей запроса на нажатие любой клавиши есть команда PAUSE, которую рекомендуется использовать перед выполнением потенциально опасных действий. Например:

ECHO Сейчас будут удалены все файлы в каталоге C:|Мои документы

ECHO Для отмены нажмите Ctrl-C

PAUSE

DEL "C:|Мои документы|.*"*

Для организации циклов используется конструкция: **FOR** %%переменная **IN (**множество**) DO** команда [параметры]

Отличие от классических языков программирования в том, что не происходит регулярного приращения счетчика, а вместо этого используется заданный список значений в параметре *множество*. Скобки здесь обязательны, внутри которых одно или несколько строковых значений, разделенных запятыми. Например:

@ECHO OFF

FOR %%i IN (Раз, Два, Три) DO ECHO %%i

напечатает следующее:

Раз

Два

Три

В качестве переменных цикла можно использовать лишь имена, состоящие из одной буквы. Весь цикл должен быть записан в одной строке, вложенные циклы не допускаются.

В параметре *множество* можно представить одну или несколько групп файлов. Например, для вывода всех файлов с расширениями *.doc и *.txt:

FOR %%f IN(C:|TEXT|.doc C:|TEXT|*.prn) DO ECHO %%f >> list.txt*

Из одного командного файла можно вызвать другой, просто указав его имя. Однако в этом случае управление назад не вернется, для этого существует команда **CALL**, с помощью которой можно в командных файлах организовать подобие подпрограмм. Например, командный файл proc.bat:

@ECHO OFF

ECHO Записываем файл %1.txt

ECHO Параметр вызова: %1 > %1.txt

можно вызвать в другом пакетной файле:

FOR %%i IN (Первый, Второй, Третий) DO CALL proc.bat %%i

в результате proc.bat вызывается 3 раза и создает 3 файла – Первый.txt, Второй.txt и Третий.txt с соответствующим текстом.

Командный файл может содержать метки, начинающиеся с двоеточия («:») и команды **GOTO** - перехода к этим меткам. Имя метки задается набором символов, следующих за двоеточием до первого пробела или конца строки. Например:

@ECHO OFF

GOTO Label1

ECHO Эта строка никогда не выполнится

:Label1

REM Продолжение выполнения

DIR

С помощью команды IF в пакетных файлах можно выполнять обработку условий 3 типов:

1) **IF [NOT]** строка1==строка2 команда

Условие считается истинным (обратите внимание на 2 знака равно) при точном совпадении обеих строк (регистр имеет значение). Строки могут быть литеральными (кавычки для них не требуются) или представлять значения переменных. Например:

IF %1%==Петя ECHO Привет, Петя !

Для предотвращения синтаксических ошибок в случае от-

сутствия переменных или параметров, рекомендуется при сравнении строк приписывать им вначале какой-нибудь символ, например:

IF -%MyVar%==-C:| ECHO OK !

2) **IF [NOT] EXIST** файл команда

Проверка существования заданного файла (кавычки для имени также не требуются, кроме длинных). Например:

IF NOT EXIST C:|autoexec.bat ECHO у вас нет файла автозагрузки !

IF EXIST "C:|Мои документы|Работа.doc" ECHO Все в порядке !

3) **IF [NOT] ERRORLEVEL** число команда

Условие считается истинным, если последняя запущенная программа или команда завершилась с кодом возврата, равным либо превышающим указанное число. Например:

```
xcopy my.txt c:| > nul
IF errorlevel 1 goto ErrOccurred
echo Копирование прошло без проблем.
goto EndBatch
:ErrOccurred
echo При выполнениеи команды возникла ошибка !
:EndBatch
```

К сожалению, в командных файлах нет возможности организовать полноценный диалог с пользователем путем ввода строк с клавиатуры. Единственное средство – команда CHOICE, которая выводит подсказку и ждет выбора пользователем варианта из указанного набора клавиш.

CHOICE [/C[:]варианты] [/N] [/S] [/T[:]c,nn] [текст]

Если в команде не задается текст, то пользователь видит на экране только подсказку. Рассмотрим подробно ключи этой команды:

Ключ	**Описание**
/C	задает варианты ответа, по умолчанию YN
CHOICE /C:ync	Yes, No, Cancel [Y, N, C]
/N	выводит только текст без вариантов ответа (клавиши для ответа работают)
CHOICE /N	

/S	учет регистра символов (по умолчанию верхний и нижний воспринимаются одинаково)
/T	*c* – символ по умолчанию, который вводится после указанной в *nn* секунд паузы

После выполнения команды CHOICE переменная ERRORLEVEL приобретает значение, равное номеру варианта ответа, что позволяет использовать команду IF для организации ветвления. Если происходит ошибка, CHOICE возвращает 255, прерывание по + или < CTRL >+ — 0. В качестве примера рассмотрим простейшее меню:

```
@ECHO OFF
ECHO Выберите режим
ECHO 1 — Простой
ECHO 2 — Расширенный
ECHO.
CHOICE /c:12 Введите пункт меню
IF ERRORLEVEL 2 goto Choice2
IF ERRORLEVEL 1 goto Choice 1
ECHO Выход из меню
GOTO Done
:Choice1
ECHO Выбран пункт 1
GOTO Done
:Choice2
ECHO Выбран пункт 2
GOTO Done
:Done
```

Особенности командных файлов в Windows NT-XP

Фактически в составе Windows NT имеются два командных интерпретатора – *command.com* во встроенной виртуальной машине MS-Dos (аналогичный по возможностям рассмотренному ранее) и специальный интерпретатор команд Windows NT – *cmd.exe (%Sytem Root%|System32)*.

Для запуска командного интерпретатора можно вызвать ярлык «Пуск/Программы/Командная строка», файловый менеджер типа FAR или загрузить cmd.exe непосредственно. При запуске cmd.exe можно использовать, как и в command.com ключи /C, /K, а также задание цвета и кодировки. Ключи /X и /Y соответ-

ственно включают и выключают расширенный режим работы интерпретатора (по умолчанию установлен).

Работа с переменными среды текущего командного окна осуществляется, как и в Windows 9x, с помощью команды SET. Естественно, изменения, которые вносятся в переменные среды этой команды, актуальны только в текущем командном окне.

Новые, весьма полезные, возможности у команды SET появляются при включении расширенной обработки команд. Теперь переменные могут рассматриваться как числа и с ними можно производить арифметические вычисления. Для этой цели имеется дополнительный ключ /A:

SET /A переменная= выражение

Использование ключа /A указывает, что стоящая справа от знака равенства строка является числовым выражением, значение которого вычисляется.

Например, если задать команду

SET /A M=1+2

то значение переменной M будет равно трем.

Обработчик выражений, входящих в команду SET, очень прост и поддерживает следующие операции, перечисленные в порядке убывания приоритета:

- группировка с помощью круглых скобок ();
- арифметические операторы умножения (*), целочисленного деления (/), остатка от деления (%);
- арифметические операторы сложения (+) и вычитания (-);
- двоичный сдвиг влево (<<) и вправо (>>);
- двоичное И (&);
- двоичное исключающее ИЛИ (л);
- двоичное ИЛИ (|);
- операторы присваивания =*, =/, =%, =+, =-, =, &=, ^=, |=, <<= и >>=;
- разделение операторов с помощью запятой (,).

При использовании любых логических или двоичных операторов необходимо заключить строку выражения в кавычки. Можно использовать префиксы для систем счисления – 0x -16, 0b — 2, 0 — 8. Любые нечисловые строки в выражении рассматриваются как имена переменных среды, значения которых преобразуются в числовой вид перед использованием. Если переменная с указанным именем не определена, вместо нее подставляется нулевое значение. Например, если переменная X не была предварительно задана, то в результате выполнения команды

SET /A N=X+5

значение N будет равно пяти.

Таким образом, применение ключа /A позволяет выполнять арифметические операции со значениями переменных среды, причем не нужно вводить знаки % для получения их значений. Кроме того, усовершенствована работа с переменными среды как со строками (кроме конкатенации – замена вхождений и выделение подстрок).

Командные файлы в Windows NT, как и в Windows 9x, являются обычными текстовыми файлами, однако для них в операционной системе зарезервированы не одно, а два расширения: bat и cmd. Новые возможности командных файлов связаны с командами SETLOCAL, ENDLOCAL, PUSHD, POPD, а также изменениями в GOTO, CALL, IF, FOR.

В Windows NT имеется возможность локализовать изменения переменных среды внутри пакетного файла, т. е. автоматически восстанавливать значения всех переменных в том виде, в каком они были до начала запуска данного файла. Команда SETLOCAL определяет начало области локальных установок переменных среды, т.е. изменения среды, внесенные после выполнения SETLOCAL, будут являться локальными относительно текущего пакетного файла. Каждая команда SETLOCAL должна иметь соответствующую команду ENDLOCAL для восстановления прежних значений переменных среды.

При работе с параметрами командного файла можно обозначить все аргументы через %*, а также использовать синтаксический анализ:

Оператор	Описание	Пример
%~Fn	Полное имя файла	%~F1=C:\TEXT\Рассказ.doc
%~Dn	Имя диска	%~D1=C:
%~Pn	Путь к файлу	%~P1=\TEXT\
%~Nn	Имя файла	%~N1=Рассказ
%~Xn	Расширение файла	%~X1=doc

Можно задавать в качестве метки перехода строку :EOF, которая передает управление в конец текущего пакетного файла. Это позволяет легко выйти из пакетного файла без определения каких-либо меток в самом его конце (**GOTO** :EOF).

В качестве адресата команды **CALL** можно использовать метки внутри текущего командного файла (аналог подпрограмм). Три вызове такой команды создается новый контекст текущего пакетного файла с заданными аргументами и управление переда-

ется на инструкцию, расположенную сразу после метки. Для выхода из такого пакетного файла необходимо два раза достичь его конца (первый выход возвращает управление на инструкцию, расположенную сразу после CALL, а второй выход завершает выполнение пакетного файла).

Дополнительно вводятся еще три варианта команды **IF**:

1) **IF** [/I] строка1 оператор_сравнения строка2 команда

В качестве операторов_сравнения используются: EQL – равно, NEQ – не равно, LSS – меньше, GTR – больше, LEQ – меньше или равно, GEQ – больше или равно. Ключ /I задает сравнение без учета регистра.

2) **IF** CMDEXTVERSION число команда

Служит для определения внутреннего номера версии расширенной обработки команд (какая на 2000 и XP — ?)

3) **IF** DEFINED переменная команда

Возвращает истинное значение, если переменная среды определена.

В Windows NT доступны еще пять разновидностей циклов **FOR**, которые обеспечивают следующие функции:

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

Выполнение заданной команды для всех подходящих имен каталогов, например получение списка всех каталогов на диске C:

FOR /D %%f IN (C:\) DO ECHO %%f

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

Организует рекурсивное выполнение заданной команды для определенного каталога, а также всех его подкаталогов. Например, для распечатки всех файлов с расширением *.txt:

FOR /R %%f IN (.txt) DO PRINT %%f*

3) **FOR /L** %переменная IN (начало,шаг,конец) DO команда [параметры]

Класический цикл с параметром (заданными началом, концом и шагом приращения). Например:

FOR /L %%f IN (1,1,5) DO ECHO %%f

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

Чтение и обработка строк из набора текстовых файлов. Ключи позволяют выделять слова и подстроки (токены) по разделителям (DELIMS) или начальным символам.

5) **FOR /F** ["ключи"] %переменная IN ('строка') DO команда [параметры]

Позволяет обработать строку вывода определенной команды (вместо набора файлов в предыдущем варианте – строка вызоыв команды в апострофах). Например, для вывода имен всех переменных среды:

FOR /F "DELIMS==" %%i IN ('SET') DO ECHO %%i

Кроме того, для переменных команды FOR разрешены также синтаксические операции (выделение имени, пути, диска — аналогично рассм. выше переменным окружения).

Команда **PUSHD** сохраняет имя текущего каталога для команды **POPD** и осуществляет переход в другой каталог. Ее синтаксис имеет вид:

PUSHD [путь | ..]

Вновь сделать текущим каталог, сохраненный командой PUSHD, можно с помощью команды POPD.

Примеры выполнения заданий

1. Пусть имеется текстовый файл protokol.txt, в котором хранится журнал обработанных файлов в следующем формате:

<имя файла> <дата> <время>

Слово дата здесь начинается в каждой строке с двадцатой позиции.

Необходимо написать командный файл, с помощью которого сделать выборку из этого файла (т. е. создать новый текстовый файл с нужной информацией) за заданный месяц (мм) и год (гггг) в файл ммгггг-.txt, сформированный файл упорядочить по дате обработки. Нужные месяц и год указать как параметры командной строки.

Решение:

Практически все нужные действия выполняются с помощью конвейеризации команд FIND и SORT:

```
@ECHO OFF
REM Проверка наличия параметров командной строки
IF -%1==- GOTO NoParam
IF -%2==- GOTO NoParam
REM Выделение нужных строк из файла protokol.txt
FIND «%1.%2" protokol.txt | SORT /+26 > %l%2.txt
GOTO End
:NoParam
ECHO Не заданы необходимые параметры командной строки!
PAUSE
```

```
:End
```

2. Написать командный файл, который будет копировать из текущего каталога все файлы с расширением txt, кроме одного файла, указанного в качестве второго параметра командной строки, в каталог, указанный первым параметром. Если имя каталога, в который должно производиться копирование, не задано, то вывести сообщение об этом и прервать выполнение файла.

Решение:

Для выполнения поставленной задачи можно перебрать в цикле все файлы с расширением txt, проверяя перед копированием имя каждого из этих файлов:

```
@ECHO OFF
REM Проверка наличия параметра командной строки
IF -%!==- GOTO NoDir
REM Копирование нужных файлов
FOR %%f IN (*.txt) DO IF NOT -%%f==-%2 COPY %%f %1
GOTO End
:NoDir
ECHO Не указан каталог для копирования!
PAUSE
:End
```

3. Задание аналогично второму упражнению. Дополнительные требования:

а) переписывать только те файлы, которые новее одноименных в каталоге — приемнике,

б) не прерывать копирование в случае возникновения ошибки,

в) записывать в файл logcopy.txt имя каждого копируемого файла и результат выполнения операции для него.

Другими словами, файл logcopy.txt должен быть примерно таким:

```
Успешно: a.txt
Ошибка : b.txt
Успешно: c.txt
```

Решение:

В этом упражнении нужно создать два пакетных файла. В основном файле работает такой же цикл FOR, как и в предыдущем упражнении, однако вместо непосредственного выполнения команды COPY здесь вызываетсякомандный файл 3_1. bat:

```
@ECHO OFF
```

REM Проверка наличия параметра командной строки
IF -%1==- GOTO NoDir
REM Вызов в цикле файла 3_l.bat для копирования нужного файла
FOR %%f IN (.txt) DO IF NOT -%%f=-%2 CALL 3_l.bat %%f %1*
GOTO End
:NoDir
ECHO Не указан каталог для копирования!
PAUSE
:End

Вызываемый в цикле файл 3_1.bat имеет следующее содержимое:

@ECHO OFF
REM Копирование файла
XCOPY %1 %2 /D /C > NUL
REM Проверка успешности копирования
IF ERRORLEVEL 0 GOTO Success
REM Запись в файл отчета информации об ошибке при копировании
ECHO Ошибка: % 1 >> logcopy.log
GOTO End
:Success
REM Запись в файл отчета информации об успешном копировании
ECHO Успешно: %1 >> logcopy.log
:End

4. Создать командный файл, который выводил бы содержимое каталога, указанного в качестве параметра командной строки, причем пользователю должна быть предоставлена возможность выбора с помощью меню устройства для вывода: на экран (информация выводится по одномуэкрану), в текстовый файл c:\catalog.txt или на принтер.

Решение:

Следующий простой пакетный файл не требует дополнительных пояснений:

@ECHO OFF
CLS
IF -%1==- GOTO NoDir
REM Вывод меню на экран
ECHO A — На экран

```
ECHO Б — В файл C:|catalog.txt
ECHO В — На принтер
KEM Вывод подсказки для ввода
CHOICE /C:АБВ Куда выводить содержимое %1
CLS
KEM Определение сделанного выбора
IF ERRORLEVEL 3 GOTO DirToPrn
IF ERRORLEVEL 2 GOTO DirToFile
IF ERRORLEVEL 1 GOTO DirToCon
ECHO Выбор не был сделан.
GOTO End
:DirToCon
DIR %1 | MORE
GOTO End
:DirToFile
DIR %1 > C:|catalog.txt
GOTO End
:DirToPrn
DIR %1 > prn
GOTO End
:NoDir
ECHO Не указан каталог для сканирования!
PAUSE
:End
```

Задания

В соответствии с номером студента по списку в журнале преподавателя разработать пакетный bat-файл. В пакетных файлах предусмотреть сообщение имени, назначения, применения и автора пакетного файла (при пустой командной строке и по ключу /?), контроль верности командной строки, наличие требуемых файлов и сохранность имени пакетного файла. Текущий каталог не изменять, если это специально не оговорено. Там, где необходимо, имена файлов указывать с полным путем и диском. С клавиатуры при работе пакетного файла вводить только числа, строковые данные выбирать либо из меню, либо передавать в командной строке.

Задания для самостоятельного выполнения.

1. Разработать пакетный файл для обновления архива. Выбор архиватора осуществляет- ся из меню. Имя архива

передается в командной строке.

2. Разработать пакетный файл для очистки подкаталога с подтверждением.

3. Разработать пакетный файл для проверки дисков (каталогов) на вирусы. Диск выбирается из меню. Имя антивирусной программы — в командной строке (drwebw.exe).

4. Разработать пакетный файл для выбора из меню на запуск одного из нескольких редакторов (notepad, ncedit и др.) для редактирования требуемого файла. Имя файла передается в командной строке.

5. Разработать пакетный файл для архивации файлов в каталогах и подкаталогах (по отдельности в каждом каталоге) и удаления архивных файлов. Имена каталога и архиватора — в командной строке.

6. Разработать пакетный файл для построения системы студенческих каталогов с запросом на создание каталогов требуемых курсов, групп и запросом максимального числа пользователей в группе. Номера курсов и шифры групп - в командной строке.

7. Разработать пакетный файл для перехода студента в личный каталог. Группа выбирается из меню, курс передается в командной строке.

8. Разработать пакетный файл для установки даты и времени (параметры – в командной строке).

9. Разработать пакетный файл для очистки студенческих каталогов: удаления файлов *.BAK, *.TMP и др. Номера очищаемых курсов передаются в командной строке.

10. Разработать пакетный файл для вывода (на экран, файл, принтер) списка файлов из каталогов студентов. Шифр группы — из командной строки, направление вывода — из меню.

11. Разработать пакетный файл для перехода в каталог студента, если он существует и его архивирования.

12. Разработать пакетный файл для вывода текстового файла на экран по страницам. Имя файла передается в командной строке.

13. Разработать пакетный файл для перезаписи файлов документов (*.doc, *.txt) из одного каталога в другой с обновлением.

14. Разработать пакетный файл для копирования всех файлов документов (*.doc, *.txt) из всех студенческих подкаталогов в директорию «Мои документы».

15. Разработать пакетный файл для проверки наличия фай-

лов документов (*.doc, *.txt) в данном подкаталоге (имя – в параметрах). В случае положительного ответа – вывести их список.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 13. «РАБОТА СО СТЕКОМ TCP/IP В СЕТЯХ WINDOWS»

Теоретическая часть

Диагностические утилиты TCP/IP.

В состав TCP/IP входят диагностические утилиты, предназначенные для проверки конфигурации стека и тестирования сетевого соединения.

Утилита	Применение
hostname	Выводит имя локального хоста. Используется без параметров.
ipconfig	Выводит значения для текущей конфигурации стека TCP/IP: IP-адрес, маску подсети, адрес шлюза по умолчанию, адреса WINS (Windows Internet Naming Service) и DNS (Domain Name System)
ping	Осуществляет проверку правильности конфигурирования TCP/IP и проверку связи с удаленным хостом.
tracert	Осуществляет проверку маршрута к удаленному компьютеру путем отправки эхо-пакетов протокола ICMP (Internet Control Message Protocol). Выводит маршрут прохождения пакетов на удаленный компьютер.
arp	Выводит для просмотра и изменения таблицу трансляции адресов, используемую протоколом разрешения адресов ARP (Address Resolution Protocol - определяет локальный адрес по IP-адресу)
route	Модифицирует таблицы маршрутизации IP. Отображает содержимое таблицы, добавляет и удаляет маршруты IP.
netstat	Выводит статистику и текущую информацию по соединению TCP/IP.
nslookup	Осуществляет проверку записей и доменных псевдонимов хостов, доменных сервисов хостов, а также информации операционной системы, путем запросов к серверам DNS.

telnet	Осуществляет соединение с другим хостом по протоколу эмуляции терминала TELNET. Используется для проверки работоспособности сетевых служб, использующих tcp-порты (например, возможности соединения с почтовым сервером по протоколам POP3 и SMTP).

1. Проверка правильности конфигурации TCP/IP с помощью ipconfig.

При устранении неисправностей и проблем в сети следует сначала проверить правильность конфигурации TCP/IP. Для этого используется утилита ipconfig.

Эта команда полезна на компьютерах, работающих с DHCP (Dynamic Host Configuration Protocol), так как дает пользователям возможность определить, какая конфигурация сети TCP/IP и какие величины были установлены с помощью DHCP.

Синтаксис:

ipconfig [/all | /renew[adapter] | /release]

Параметры:

all - выдает весь список параметров. Без этого ключа отображается только IP-адрес, маска и шлюз по умолчанию;

renew[adapter] - обновляет параметры конфигурации DHCP для указанного сетевого адаптера;

release[adapter] - освобождает выделенный DHCP IP-адрес;

adapter – имя сетевого адаптера;

displaydns - выводит информацию о содержимом локального кэша клиента DNS, используемого для разрешения доменных имен.

Таким образом, утилита ipconfig позволяет выяснить, инициализирована ли конфигурация и не дублируются ли IP-адреса:

- если конфигурация инициализирована, то появляется IP-адрес, маска, шлюз;
- если IP-адреса дублируются, то маска сети будет 0.0.0.0;
- если при использовании DHCP компьютер не смог получить IP-адрес, то он будет равен 0.0.0.0 .

2. Тестирование связи с использованием утилиты *ping*.

Утилита ping (Packet Internet Grouper) используется для проверки конфигурирования TCP/IP и диагностики ошибок соединения. Она определяет доступность и функционирование конкретного хоста. Использование ping лучший способ проверки того, что между локальным компьютером и сетевым хостом существует маршрут. Хостом называется любое сетевое устройство

(компьютер, маршрутизатор), обменивающееся информацией с другими сетевыми устройствами по TCP/IP.

Команда ping проверяет соединение с удаленным хостом путем посылки к этому хосту эхо-пакетов ICMP и прослушивания эхо-ответов. Ping ожидает каждый посланный пакет и печатает количество переданных и принятых пакетов. Каждый принятый пакет проверяется в соответствии с переданным сообщением. Если связь между хостами плохая, из сообщений ping станет ясно, сколько пакетов потеряно.

По умолчанию передается 4 эхо-пакета длиной 32 байта (возможны и другие варианты значения по умолчанию) - периодическая последовательность символов алфавита в верхнем регистре. Ping позволяет изменить размер и количество пакетов, указать, следует ли записывать маршрут, который она использует, какую величину времени жизни (ttl) устанавливать, можно ли фрагментировать пакет и т. д.. При получении ответа в поле time указывается, за какое время (в миллисекундах) посланный пакет доходит до удаленного хоста и возвращается назад. Так как значение по умолчанию для ожидания отклика равно 1 секунде, то все значения данного поля будут меньше 1000 миллисекунд. Если вы получаете сообщение «Request time out» (Превышен интервал ожидания), то, возможно, если увеличить время ожидания отклика, пакет дойдет до удаленного хоста. Это можно сделать с помощью ключа –w.

Ping можно использовать для тестирования как имени хоста (DNS или NetBIOS), так и его IP-адреса. Если ping с IP-адресом выполнилась успешно, а с именем – неудачно, это значит, что проблема заключается в распознавании соответствия адреса и имени, а не в сетевом соединении.

Утилита ping используется следующими способами:

1) Для проверки того, что TCP/IP установлен и правильно сконфигурирован на локальном компьютере, в команде ping задается адрес петли обратной связи (loopback address): ping 127.0.0.1

Если тест успешно пройден, то вы получите следующий ответ:

Ответ от 127.0.0.1: число байт=32 время<1мс TTL=128
Ответ от 127.0.0.1: число байт=32 время<1мс TTL=128
Ответ от 127.0.0.1: число байт=32 время<1мс TTL=128
Ответ от 127.0.0.1: число байт=32 время<1мс TTL=128

2) Чтобы убедиться в том, что компьютер правильно добавлен в сеть и IP-адрес не дублируется, используется IP-адрес локального компьютера:

ping IP-адрес_локального_хоста

3) Чтобы проверить, что шлюз по умолчанию функционирует и что можно установить соединение с любым локальным хостом в локальной сети, задается IP-адрес шлюза по умолчанию:

ping IP-адрес_шлюза

4) Для проверки возможности установления соединения через маршрутизатор в команде ping задается IP-адрес удаленного хоста:

ping IP-адрес_удаленного хоста

Синтаксис:

ping [-t] [-a] [-n count] [-l length] [-f] [-i ttl] [-v tos] [-r count] [-s count] [[-j host-list] |

[-k host-list]] [-w timeout] destination-list

Параметры:

-t выполняет команду ping до прерывания. Control-Break - посмотреть статистику и продолжить. Control-C - прервать выполнение команды;

-a позволяет определить доменное имя удаленного компьютера по его IP-адресу;

-n count посылает количество пакетов ECHO, указанное параметром count;

-l length посылает пакеты длиной length байт (максимальная длина 8192 байта);

-f посылает пакет с установленным флагом «не фрагментировать». Этот пакет не будет фрагментироваться на маршрутизаторах по пути своего следования;

-i ttl устанавливает время жизни пакета в величину ttl (каждый маршрутизатор уменьшает ttl на единицу);

-v tos устанавливает тип поля «сервис» в величину tos;

-r count записывает путь выходящего пакета и возвращающегося пакета в поле записи пути. Count - от 1 до 9 хостов;

-s count позволяет ограничить количество переходов из одной подсети в другую (хопов). Count задает максимально возможное количество хопов;

-j host-list направляет пакеты с помощью списка хостов, определенного параметром host-list. Последовательные хосты могут быть отделены промежуточными маршрутизаторами (гибкая статическая маршрутизация). Максимальное количество хостов в списке, позволенное IP, равно 9;

-k host-list направляет пакеты через список хостов, определенный в host-list. Последовательные хосты не могут быть разделены промежуточными маршрутизаторами (жесткая статическая

маршрутизация). Максимальное количество хостов – 9;

-w timeout указывает время ожидания (timeout) ответа от удаленного хоста в миллисекундах (по умолчанию – 1сек);

destination-list указывает удаленный хост, к которому надо направить пакеты ping.

Пример использования утилиты ping:

C:\WINDOWS>ping –n 10

Обмен пакетами с [205.188.247.65] по 32 байт:

Ответ от 205.188.247.65: число байт=32 время=194мс TTL=48

Ответ от 205.188.247.65: число байт=32 время=240мс TTL=48

Ответ от 205.188.247.65: число байт=32 время=173мс TTL=48

Ответ от 205.188.247.65: число байт=32 время=250мс TTL=48

Ответ от 205.188.247.65: число байт=32 время=187мс TTL=48

Ответ от 205.188.247.65: число байт=32 время=239мс TTL=48

Ответ от 205.188.247.65: число байт=32 время=263мс TTL=48

Ответ от 205.188.247.65: число байт=32 время=230мс TTL=48

Ответ от 205.188.247.65: число байт=32 время=185мс TTL=48

Ответ от 205.188.247.65: число байт=32 время=406мс TTL=48

Статистика Ping для 205.188.247.65:

Пакетов: послано = 10, получено = 10, потеряно = 0 (0% потерь)

Приблизительное время передачи и приема:

Наименьшее = 173мс, наибольшее = 406мс, среднее =236мс

В случае невозможности проверить доступность хоста утилита выводит информацию об ошибке. Ниже приведен пример ответа утилиты ping при попытке послать запрос на несуществующий хост.

Обмен пакетами с 172.16.6.21 по 32 байт:

Превышен интервал ожидания для запроса.

Превышен интервал ожидания для запроса.

Превышен интервал ожидания для запроса.

Превышен интервал ожидания для запроса.

Статистика Ping для 172.16.6.21:

Пакетов: отправлено = 4, получено = 0, потеряно = 4 (100% потерь),

Приблизительное время передачи и приема:

наименьшее = 0мс, наибольшее = 0мс, среднее = 0мс.

Утилита сообщает не об отсутствии хоста, а о том, что за отведенное время не был получен ответ на посланный запрос. Причиной этого не обязательно является отсутствие хоста в сети. Проблема может крыться в сбоях связи, перегрузке или неправильной настройке маршрутизаторов и т. п. Ошибка «сеть недоступна» (network unreachable) прямо указывает на проблемы маршрутизации.

3. Изучение маршрута между сетевыми соединениями с помощью утилиты *tracert*.

Tracert - это утилита трассировки маршрута. Она использует поле TTL (time-to-live, время жизни) пакета IP и сообщения об ошибках ICMP для определения маршрута от одного хоста до другого.

Утилита tracert может быть более содержательной и удобной, чем ping, особенно в тех случаях, когда удаленный хост недостижим. с помощью нее можно определить район проблем со связью (у Internet-провайдера, в опорной сети, в сети удаленного хоста) по тому, насколько далеко будет отслежен маршрут. Если возникли проблемы, то утилита выводит на экран звездочки (*), либо сообщения типа «Destination net unreachable», «Destination host unreachable», «Request time out», «Time Exeeded».

Утилита tracert работает следующим образом: посылается по 3 пробных эхо-пакета на каждый хост, через который проходит маршрут до удаленного хоста. На экран при этом выводится время ожидания ответа на каждый пакет (Его можно изменить с помощью параметра - w). Пакеты посылаются с различными величинами времени жизни. Каждый маршрутизатор, встречающийся по пути, перед перенаправлением пакета уменьшает величину TTL на единицу. Таким образом, время жизни является счетчиком точек промежуточной доставки (хопов). Когда время жизни пакета достигнет нуля, предполагается, что маршрутизатор пошлет в компьютер-источник сообщение ICMP “Time Exeeded” (Время истекло). Маршрут определяется путем посылки первого эхо-пакета с TTL=1. Затем TTL увеличивается на 1 в каждом последующем пакете до тех пор, пока пакет не достигнет удаленного хоста, либо будет достигнута максимально возможная величина TTL (по

умолчанию 30, задается с помощью параметра - h).

Маршрут определяется путем изучения сообщений ICMP, которые присылаются обратно промежуточными маршрутизаторами.

Примечание: некоторые маршрутизаторы просто молча уничтожают пакеты с истекшим TTL и не будут видны утилите tracert.

Синтаксис:

tracert [-d] [-h maximum_hops] [-j host-list] [-w timeout] имя_целевого_хоста

Параметры:

-d указывает, что не нужно распознавать адреса для имен хостов;

-h maximum_hops указывает максимальное число хопов для того, чтобы искать цель;

-j host-list указывает нежесткую статическую маршрутизацию в соответствии с host-list;

-w timeout указывает, что нужно ожидать ответ на каждый эхо-пакет заданное число мсек.

4. Утилита *arp*.

Основная задача протокола ARP – трансляция IP-адресов в соответствующие локальные адреса. Для этого ARP-протокол использует информацию из ARP-таблицы (ARP-кэша). Если необходимая запись в таблице не найдена, то протокол ARP отправляет широковещательный запрос ко всем компьютерам локальной подсети, пытаясь найти владельца данного IP-адреса. В кэше могут содержаться два типа записей: статические и динамические. Статические записи вводятся вручную и хранятся в кэше постоянно. Динамические записи помещаются в кэш в результате выполнения широковещательных запросов. Для них существует понятие времени жизни. Если в течение определенного времени (по умолчанию 2 мин.) запись не была востребована, то она удаляется из кэша.

Синтаксис:

arp [-s inet_addr eth_addr] | [-d inet_addr] | [-a]

Параметры:

-s занесение в кэш статических записей;

-d удаление из кэша записи для определенного IP-адреса;

-a просмотр содержимого кэша для всех сетевых адаптеров локального компьютера;

inet_addr - IP-адрес;

eth_addr - MAC-адрес.

5. Утилита route.

Утилита route предназначена для работы с локальной таблицей маршрутизации. Она имеет следующий

Синтаксис: route [-f] [-p] [команда [узел] [MASK маска] [шлюз] [METRIC метрика] [IF интерфейс]]

Параметры:

-f Очистка таблицы маршрутизации.

-p При указании совместно с командой ADD создает постоянную запись, которая сохраняется после перезагрузки компьютера. По умолчанию записи таблицы маршрутов не сохраняются при перезагрузке.

команда одна из четырех команд:

PRINT - вывод информации о маршруте;

ADD - добавление маршрута;

DELETE - удаление маршрута;

CHANGE - изменение маршрута.

узел адресуемый узел

маска маска подсети; по умолчанию используется маска 255.255.255.255

шлюз адрес шлюза

метрика метрика маршрута;

интерфейс идентификатор интерфейса, который будет использован для пересылки пакета

Для команд PRINT и DELETE возможно использование символов подстановки при указании адресуемого узла или шлюза. Параметр шлюза для этих команд может быть опущен.

При добавлении и изменении маршрутов утилита route осуществляет проверку введенной информации на соответствие условию (УЗЕЛ & МАСКА) == УЗЕЛ. Если это условие не выполняется, то утилита выдает сообщение об ошибке и не добавляет или не изменяет маршрут.

Утилита осуществляет поиск имен сетей в файле networks. Поиск имен шлюзов осуществляется в файле hosts. Оба файла расположены в папке %systemroot%\system32\drivers\etc. Наличие и заполнение этих файлов не обязательно для нормального функционирования утилиты route и работы маршрутизации.

Хотя в большинстве случаев на рабочей станции это не требуется, можно вручную редактировать таблицы маршрутизации.

Пример использования утилиты route:

Добавление статического маршрута: route add 172.16.6.0 MASK 255.255.25METRIC 1 IF 0x1000003

6. Утилита *netstat*.

Утилита netstat позволяет получить статическую информацию по некоторым из протоколов стека (TCP, UDP, IP, ICMP), а также выводит сведения о текущих сетевых соединениях. Особенно она полезна на брандмауэрах, с ее помощью можно обнаружить нарушения безопасности периметра сети.

Синтаксис:

netstat [-a] [-e] [-n] [-s] [-p protocol] [-r]

Параметры:

-a выводит перечень всех сетевых соединений и прослушивающихся портов локального компьютера;

-e выводит статистику для Ethernet-интерфейсов (например, количество полученных и отправленных байт);

-n выводит информацию по всем текущим соединениям (например, TCP) для всех сетевых интерфейсов локального компьютера. Для каждого соединения выводится информация об IP-адресах локального и удаленного интерфейсов вместе с номерами используемых портов;

-s выводит статистическую информацию для протоколов UDP, TCP, ICMP, IP. Ключ «/more» позволяет просмотреть информацию постранично;

-r выводит содержимое таблицы маршрутизации.

7. Утилита *nslookup*.

Утилита nslookup предназначена для диагностики службы DNS, в простейшем случае - для выполнения запросов к DNS-серверам на разрешение имен в IP-адреса. В общем случае утилита позволяет просмотреть любые записи DNS-сервера:

A – каноническое имя узла, устанавливает соответствие доменного имени ip-адресу.

SOA – начало полномочий, начальная запись, единственная для зоны;

MX – почтовые серверы (хосты, принимающие почту для заданного домена);

NS – серверы имен (содержит авторитетные DNS-серверы для зоны);

PTR – указатель (служит для обратного преобразования ip-адреса в символьное имя хоста)

и т. д.

Утилита nslookup достаточно сложна и содержит свой собственный командный интерпретатор.

В простейшем случае (без входа в командный режим) утилита nslookup имеет следующий

Синтаксис: nslookup хост [сервер]

Параметры:

Хост DNS-имя хоста, которое должно быть преобразовано в IP-адрес.

Сервер Адрес DNS-сервера, который будет использоваться для разрешения имени. Если этот параметр опущен, то будут последовательно использованы адреса DNS-серверов из параметров настройки протокола TCP/IP.

Примеры использования утилиты nslookup:

1. Получение списка серверов имен для домена ***** без входа в командный режим (с использованием ключей).

C:\> nslookup - type=ns *****
Server: dns01.catv. *****
Address: 217.10.44.35
Non-authoritative answer:
***** nameserver = *****
***** nameserver = *****
***** nameserver = *****
***** nameserver = *****
***** internet address = 213.180.199.34
***** internet address = 213.180.204.1

2. Получение записи SOA домена ***** с авторитетного сервера с использование командного интерпретатора nslookup.

C:\>nslookup
Default Server: dns04.catv. *****
Address: 217.10.39.4
> set type=SOA
> server *****
Default Server: *****
Address: 213.180.199.34
> *****
Server: *****
Address: 213.180.193.1
>*****
primary name server = *****
responsible mail addr = sysadmin. yandex-team. r
serial =
refresh = 1mins)
retry = mins)
expire = 2592days)
default TTL = mins)
***** nameserver = *****

***** nameserver = *****
***** nameserver = *****
***** nameserver = *****
***** internet address = 213.180.193.1
***** internet address = 213.180.199.34
***** internet address = 77.88.19.60
***** internet address = 213.180.204.1
> exit

3. Получение адреса почтового сервера для домена *****.

C:\ >nslookup
Default Server: dns01.catv. *****
Address: 217.10.44.35
> set q=mx
> *****
Server: dns01.catv. *****
Address: 217.10.44.35
Non-authoritative answer:
***** MX preference = 10, mail exchanger = *****
***** MX preference = 10, mail exchanger = *****
***** MX preference = 10, mail exchanger = *****
***** nameserver = *****
***** nameserver = *****
***** nameserver = *****
***** nameserver = *****
***** internet address = 77.88.21.89
***** internet address = 93.158.134.89
***** internet address = 213.180.204.89
***** internet address = 213.180.199.34
***** internet address = 77.88.19.60
***** internet address = 213.180.204.1
>

Указав ключ type=any, можно получить все записи о узле или домене. Ключи querytype, t, q эквивалентны type.

8. Утилита *telnet*.

Утилита telnet (TELecommunication NETwork) реализует клиентскую часть сетевого протокола telnet, организующего текстовый интерфейс по сети (при помощи транспортного протокола TCP).

Исторически Telnet служил для удалённого доступа к интерфейсу командной строки операционных систем. Впоследствии его стали использовать для прочих текстовых интерфейсов, вплоть до игр MUD и анимированного ASCII-art. Теоретически,

даже обе стороны протокола могут являться программами, а не человеком.

Иногда клиенты telnet используются для доступа к другим протоколам на основе транспорта TCP.

Протокол telnet используется в управляющем соединении FTP, т.е. заходить на сервер командой telnet ftp. ftp для выполнения отладки и экспериментов не только возможно, но и правильно (в отличие от применения клиентов telnet для доступа к HTTP, IRC и большинству других протоколов).

В протоколе не предусмотрело ни шифрования, ни проверки подлинности данных. Поэтому он уязвим для любого вида атак на TCP. Для функциональности удалённого доступа к системе в настоящее время применяется сетевой протокол SSH (особенно его версия 2), при создании которого упор делался именно на вопросы безопасности. Следует иметь в виду, что сессия telnet обладает крайне низкой защищенностью, если только не осуществляется в полностью контролируемой сети или с применением защиты на сетевом уровне (различные реализации виртуальных частных сетей). По причине ненадёжности от telnet как средства управления операционными системами давно отказались.

Тем не менее, клиент telnet пригоден для осуществления ручного доступа (например, в целях отладки) к таким протоколам прикладного уровня как HTTP, IRC, SMTP, POP3 и прочим текст-ориентированным протоколам на основе транспорта TCP.

По умолчанию (если порт не задан), telnet использует порт 23.

Синтаксис:

telnet имя_узла номер_порта

Примеры использования утилиты telnet:

1) Доступ к почтовому серверу по протоколу POP3 (проверка работоспособности почтового ящика).

Введите: telnet имя_почтового_сервера 110

Ответ сервера:

+OK Hello there.

В качестве имени пользователя введите свой адрес электронной почты:

user *****@***ru

Ответ сервера:

+OK Password required.

Введите пароль для этого почтового ящика:

pass *пароль*

Ответ сервера:

+OK logged in.
Для выхода введите: quit
+OK Bye-bye
2) Проверка доступа к smtp-серверу.
Введите:
telnet имя_почтового_сервера 25

Если в результате Вы получите сообщение, начинающееся с цифры 2, то у Вас есть доступ к smtp-серверу, в противном случае можно судить об ошибке.

Задания для самостоятельного выполнения

1. Изучите методические указания к лабораторной работе.
2. Выполните упражнения.
3. Оформите отчет по лабораторной работе, описав выполнение упражнений и дав краткие ответы на контрольные вопросы.

Упражнение 1. Получение справочной информации по командам.

Выведите на экран справочную информацию по всем рассмотренным утилитам (см. таблицу п.1). Для этого в командной строке введите имя утилиты без параметров или с /?. Для получения справочной информации по nslookup необходимо войти в командный режим, набрав nslookup без параметров, и ввести команду help.

Изучите ключи, используемые при запуске утилит.

Упражнение 2. Получение имени хоста.

Выведите на экран имя локального хоста с помощью команды hostname.

Упражнение 3. Изучение утилиты ipconfig.

Проверьте конфигурацию TCP/IP с помощью утилиты ipconfig.

Заполните таблицу:

Имя хоста	
IP-адрес	
Маска подсети	
Основной шлюз	
Используется ли DHCP (адрес DHCP-сервера)	
Описание адаптера	

Физический адрес сетевого адаптера	
Адрес DNS-сервера	
Адрес WINS-сервера	

Упражнение 4. Тестирование связи с помощью утилиты ping.

1. Проверьте правильность установки и конфигурирования TCP/IP на локальном компьютере.

2. Проверьте, правильно ли добавлен в сеть локальный компьютер и не дублируется ли IP-адрес.

3. Проверьте функционирование шлюза по умолчанию, послав 5 эхо-пакетов длиной 64 байта.

4. Проверьте возможность установления соединения с удаленным хостом.

5. С помощью команды ping проверьте перечисленные ниже адреса и для каждого из них отметьте время отклика. Попробуйте изменить параметры команды ping таким образом, чтобы увеличилось время отклика. Определите IP-адреса узлов.

a) *****

b) router. *****

c) любой узел из локальной сети

Упражнение 5. Определение пути IP-пакета.

С помощью команды tracert проверьте для перечисленных ниже адресов, через какие промежуточные узлы идет сигнал. Время жизни установить равным 10. Отметьте их:

a) 195.82.146.114

b) *****

c) 213.247.189.211

Упражнение 6: Просмотр ARP-кэша.

С помощью утилиты arp просмотрите ARP-таблицу локального компьютера.

Внести в кэш локального компьютера любую статическую запись.

Упражнение 7: Просмотр локальной таблицы маршрутизации.

С помощью утилиты route просмотреть локальную таблицу маршрутизации.

Упражнение 8. Получение информации о текущих сетевых соединениях и протоколах стека TCP/IP.

С помощью утилиты netstat выведите перечень сетевых соединений и статистическую информацию для протоколов UDP, TCP, ICMP, IP.

Упражнение 9. Получение DNS-информации с помощью nslookup.

1) Узнайте ip-адреса узлов:

*****

photo. *****

sova. *****

wiki. *****

share. *****

2) Узнайте авторитетные (компетентные) сервера для этих узлов.

3) Получите запись SOA с одного из этих серверов для домена *****.

Упражнение 10. Диагностика tcp-соединений с помощью утилиты telnet.

1) Проверить, принимает ли хост share. ***** подключения по SMB (445 порт).

2) Присоединиться к 4899 порту хоста 213.247.189.211.

3) Узнать, какой почтовый сервер использует Майкрософт (использовать nslookup + telnet).

Контрольные вопросы

1. Раскрыть термины: хост, шлюз, хоп, время жизни пакета, маршрут, маска сети, авторитетный/неавторитетный (компетентный) DNS-сервер, порт TCP, петля обратной связи, время отклика.

2. Какие утилиты можно использовать для проверки правильности конфигурирования TCP/IP?

3. Каким образом каманда ping проверяет соединение с удаленным хостом?

4. Сколько промежуточных маршрутизаторов сможет пройти IP-пакет, если его время жизни равно 30?

5. Как работает утилита tracert?

6. Каково назначение протокола ARP?

7. Как утилита ping разрешает имена узлов в ip-адреса (и наоборот)?

8. Какие могут быть причины неудачного завершения ping и tracert? (превышен интервал ожидания для запроса, сеть недоступна, превышен срок жизни при передаче пакета).

9. Объяснить, каким образом при неудачной проверке маршрута до хоста 213.247.189.211, к нему возможно подключиться telnet'ом.

10. Всегда ли можно узнать символьное имя узла по его ip-

адресу?

11. Какой тип записи запрашивает у DNS-сервера простейшая форма nslookup?

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 14. «КОМАНДНЫЕ ФАЙЛЫ ОПЕРАЦИОННОЙ СИСТЕМЫ WINDOWS»

Теоретическая часть

Цель занятия: научиться пользоваться основными командами операционной системы Windows.

Задание: составить конспект основных структур командного файла, составить командный файл для загрузки системы в минимальной конфигурации, следуя инструкциям описания.

Общие сведения о командном процессоре Windows

Командные файлы (скрипты, сценарии, батники) - это обычные текстовые файлы с расширением .bat или .cmd, строки которых представляют собой специальные команды или имена исполняемых файлов. Строки командных файлов обрабатываются специальной программой - командным процессором операционной системы, часто называемым интерпретатором команд. Для операционных систем DOS и Windows9X в качестве интерпретатора команд используетсяcommand.com, для Windows NT и старше - cmd.exe.

Строки командных файлов могут содержать специфические команды самого процессора команд (FOR, ECHO, REM и т.п.) или имена исполняемых модулей (net.exe, regedit.exe, sc.exe) Командный процессор может быть запущен в интерактивном режиме через Пуск - Выполнить - CMD.EXE. В данном режиме, вы увидите окно консоли с приглашением к вводу команд. Возможный список большинства консольных команд можно получить введя: HELP. Справочную информацию по конкретной команде можно получить, указав ее название в качестве параметра команды HELP: HELP Имя команды.

Если работа осуществляется в русифицированной версии Windows, то учтите, что в среде командного процессора символы национального алфавита используются в DOS-кодировке. Для переключения между кодовыми страницами Windows и DOS используется команда

CHCP номер страницы:

CHCP 866 - использовать кодовую страницу 866 (DOS);

CHCP 1251 - использовать кодовую страницу 1251 (WINDOWS).

Для просмотра и редактирования командных файлов, содержащих символы русского алфавита нужно использовать редактор с поддержкой DOS-кодировки. Если вы используете стандартное приложение "Блокнот" (notepad.exe), то для правильного отображения символов русского алфавита нужно выбрать шрифт Terminal, с помощью меню Правка - Шрифт...

Внешний вид окна CMD.EXE (консоли Windows) можно изменить с помощью команды COLOR.

В качестве аргументов для команды используются 2 шестнадцатеричные цифры, задающие цвет фона и цвет символа.

COLOR F0 - черные символы на белом фоне.

COLOR 0E - светло-желтые символы на черном фоне.

HELP COLOR - подсказка для команды COLOR.

Работа с командным процессором предполагает использование двух устройств - устройства ввода (клавиатуры) и устройства вывода (дисплей). Однако, имеется возможность изменить стандартно используемые устройства ввода-вывода с помощью специальных символов - символов перенаправления:

> - перенаправление вывода

< - перенаправление ввода

Для вывода справки не на экран а, например, в файл с именем help.txt, можно использовать следующую команду:

HELP > help.txt.

При выполнении данной команды, в текущем каталоге будет создан файл с именем help.txt, содержимым которого будет результат вывода команды HELP. Если файл help.txt существовал на момент выполнения команды, его содержимое будет перезаписано. Для того чтобы дописать данные в конец существующего файла, используют удвоение символа перенаправления вывода - ">>". Например:

HELP GOTO > myhelp.txt - в файл myhelp.txt будет выдана справка по использованию команду GOTO;

HELP COLOR >> myhelp.txt - в конец файла myhelp.txt будет дописана справка по использованию команды COLOR.

Простейший пример перенаправления ввода:

cmd.exe < commands.txt - командный процессор не будет ожидать ввода команд с клавиатуры, а считает их из файла commands.txt.

При запуске командного процессора можно указать конкретную команду в качестве аргумента командной строки:

cmd.exe /C HELP FOR - выполнить команду HELP FOR и завершиться (ключ /C);

cmd.exe /K HELP FOR - выполнить команду HELP FOR и перейти в режим ожидания дальнейшего ввода команд (ключ /K).

Подробную справку по использованию cmd.exe можно получить, введя в качестве аргумента ключ /?

cmd.exe /?

Кроме символов перенаправления ввода-вывода в командной строке могут использоваться символы объединения команд - && и || :

cmd.exe /C "HELP IF > nul" && Echo HELP Executed || Echo HELP Not Executed - выполнить команду HELP IF и при успешном результате выполнить команду Echo HELP Executed, а при неуспешном - Echo HELP Not Executed. Команды, объединяемые для выполнения с помощью конструкции && , не нужно заключать в двойные кавычки. Выполнение строки cmd.exe /C "HELP IF > nul" && Echo HELP Executed || Echo HELP Not Executed завершится сообщением HELP Executed, а выполнение cmd.exe /C "HELP uIF > nul" && Echo HELP Executed || Echo HELP Not Executed где неверно задан аргумент команды HELP (uIF), завершится сообщением HELP Not Executed.

Файлы с расширением .bat или .cmd в среде Windows стандартно открываются командным процессором аналогично примеру, где список команд считывается не с устройства ввода, а из текстового файла.

Использование переменных в командных файлах

Существует такое понятие, как переменные окружения (environments) - это переменные, значения которых характеризуют среду, в которой выполняются команда или пакетный файл. Иногда их называют переменными среды. Принимаемые значения этих переменных формируются при загрузке, регистрации пользователя в системе, старте или завершении некоторых приложений, и, кроме того, могут быть заданы с помощью специальной команды SET:

SET переменная=строка

где: переменная - имя переменной среды;

строка - строка символов, присваиваемая указанной переменной.

Например, командная строка

SET myname=Vasya

создает переменную myname, принимающую значение Vasya.

Значение, присвоенное какой-либо переменной, доступно

для обработки в командных файлах, при использовании ее имени, заключенного в знаки процента - % . Например команда выдачи текста на дисплей ECHO в виде:

ECHO date - выведет на экран слово "date", а команда ECHO %date% выведет на экран значение переменной date - текущую дату в формате операционной системы.

С помощью команды SET обычно задается и модифицируется путь поиска исполняемых программ - переменная окружения PATH:

SET PATH=C:\Windows;C:\windows\system32

После выполнения данной команды, поиск исполняемых файлов будет выполняться в каталоге C:\Windows, и, если результат неуспешен, в C:\windows\system32.

Допустим, необходимо выполнить программу myedit.exe, размещенную в каталоге C:\NewProgs. Если в командной строке не задан полный путь, а только имя исполняемого файла - myedit.exe, то сначала будет выполняться поиск файла myedit.exe в текущем каталоге, и если он не будет найден - в каталогах, список которых задается значением переменной PATH. Символ «;» является разделителем элементов в списке путей поиска. Если в приведенном примере текущим каталогом не является C:\NewProgs, и в остальных каталогах, заданных значением переменной PATH, нет исполняемого файла myedit.exe, то запуск приложения myedit.exe завершится ошибкой. Однако если есть необходимость его запуска без указания полного пути и при любом значении текущего каталога, можно модифицировать значение переменной PATH.

Команда SET PATH=C:\NewProgs;%path% изменит текущее значение PATH, добавив каталог C:\NewProgs в начало списка. Выполнение команды SET без параметров позволяет получить текущие значения переменных окружения:

NUMBER_OF_PROCESSORS=1 - количество процессоров

OS=Windows_NT- тип ОС

Path=C:\WINDOWS\system32;C:\WINDOWS;C:\Program Files\Far - путь поиска исполняемых файлов.

PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH - расширения для исполняемых файлов.

PROCESSOR_ARCHITECTURE=x86 - архитектура процессора.

PROCESSOR_IDENTIFIER=x86 Family 6 Model 8 Stepping 1, AuthenticAMD - идентификатор процессора.

PROCESSOR_LEVEL=6 - уровень (номер модели) процессора.

PROCESSOR_REVISION=0801 - версия процессора.

ProgramFiles=C:\Program Files - путь к папке "Program Files"

PROMPT=PG - формат приглашения командной строки $P - путь для текущего каталога $G - знак ">".

SystemDrive=C: - буква системного диска.

SystemRoot=C:\WINDOWS - каталог ОС Windows.

Значение некоторых переменных по команде SET не выдаются. В основном, это переменные, принимаемые значения которых динамически изменяются:

%CD% - Принимает значение строки текущего каталога.

%DATE% - Принимает значение текущей даты.

%TIME% - Принимает значение текущего времени.

%RANDOM% - Принимает значение случайного десятичного числа в диапазоне 1 -32767.

%ERRORLEVEL% - Принимает текущее значение кода завершения задачи ERRORLEVEL

%CMDEXTVERSION% - Принимает значение версии командного процессора CMD.EXE для расширенной обработки команд.

%CMDCMDLINE% - Принимает значение строки, которая вызвала командный процессор.

Для просмотра действующего значения какой-либо переменной обычно используется команда: ECHO %переменная%:

ECHO %CD% - отобразить имя текущего каталога;

ECHO %TIME% - отобразить текущее время;

ECHO %ERRORLEVEL% - отобразить результат выполнения предыдущей команды.

Значения, принимаемые переменными окружения, могут быть расширены с помощью специального признака - символа" ~ ", что получить частичное значение (расширение переменной), или изменить его заменой какой-либо части. Примеры использования расширений переменных рассмотрены ниже.

Передача параметров командному файлу.

Очень полезной особенностью работы с командными файлами является возможность передавать параметры командной строки и использовать их значения в операциях внутри самого командного файла.

BAT-файл параметр1 параметр2 ... параметрN.

В самом командном файле первый параметр будет доступен как переменная %1, второй - %2 и т.п. Путь и имя самого командного файла доступно как переменная %0. Для примера создадим командный файл, задачей которого будет выдача на экран значений введенных при его запуске параметров командной стро-

ки. Для вывода текста на экран используется командаECHO текст, однако если "текст" заменить на %0, - то будет выдано имя командного файла, %1 - первый аргумент, заданный в строке запуска, %2 - второй и т.д.

Создаем, например, командный файл params.bat следующего содержания:

```
echo off echo Это командный файл %0
echo Первый параметр=%1
echo Второй параметр=%2
echo Третий параметр = %3
```

и запускаем его на выполнение следующей командой:

```
params.bat FIRST second "two words"
```

параметры содержащие пробелы, нужно заключать в двойные кавычки.

В первой строке командного файла используется команда "echo off" для того, чтобы обрабатываемые командным процессором строки не выдавались на экран.

Для проверки наличия каких-либо входных параметров, передаваемых командному файлу, можно проверить, является ли значение переменной %1 пустым:

```
if "%1" EQU "" goto error
....
...
:error
```

Переходы и метки

В командных файлах можно использовать команды условного перехода, меняющие логику их работы в зависимости от выполнения определенных условий. Для иллюстрации приемов использования условных переходов создадим командный файл, целью которого будет присвоение заранее определенной буквы для съемных носителей, в качестве которых будут использоваться флэш-диски. Условия таковы - есть 2 флэш-диска, один из которых должен быть виден в проводнике как диск X: а второй - как диск Y: независимо от того, в какой порт USB они подключены и какие буквы присвоены им операционной системой. Будем считать, что реальные диски могут быть подключены как F: или G: Опознавание дисков будем выполнять по наличию файла с определенным именем (лучше такой файл сделать скрытым в корневом каталоге и назвать его как-нибудь необычно):

```
Flashd1.let - на первом диске
Flashd2.let - на втором диске
```

Таким образом, задача командного файла заключается в том, чтобы проверить наличие на сменных дисках F: иG: файлов Flashd1.let или Flashd2.let и, в зависимости от того, какой из них присутствует, присвоить диску букву X: или Y:

Для поиска файла на диске воспользуемся командой IF EXIST:

IF EXIST имя_файла команда.

В качестве команды проще всего воспользоваться SUBST, сопоставляющей имя диска и каталог. SUBST X: C:\ - - создать виртуальный диск X:, содержимым которого будет корневой каталог диска C:. Для решения задачи, создаем командный файл, например setletter.bat, следующего содержания:

```
@ECHO OFF
IF EXIST G:\flashd1.let SUBST X: G:\
IF EXIST F:\flashd1.let SUBST X: F:\
IF EXIST G:\flashd2.let SUBST Y: G:\
IF EXIST F:\flashd2.let SUBST Y: F:\
```

После выполнения этого командного файла у вас появятся диски X: и Y:. Однако, если такой файл выполнить повторно, команда SUBST выдаст сообщение об ошибке - ведь диски X: и Y: уже существуют. Поэтому, желательно обойти выполнение SUBST, если виртуальные диски X: и Y: уже созданы, или удалять их, используя SUBST с параметром -d перед подключением. Попробуйте изменить командный файл setletter.bat с использованием команды перехода GOTO, осуществляющей передачу управления строке пакетного файла на указанную метку: GOTO метка.

В качестве метки используется строка символов, начинающаяся с двоеточия. Сделаем измене-ния в нашем командном файле, чтобы не возникало сообщений об ошибке:

```
@ECHO OFF
REM если не существует X: - то перейдем на метку SETX
IF NOT EXIST X:\ GOTO SETX
REM если существует X: - перейдем на проверку наличия Y:
GOTO TESTY
:SETX
IF EXIST G:\flashd1.let SUBST X: G:\
IF EXIST F:\flashd1.let SUBST X: F:\
:TESTY
REM если Y: существует - завершим командный файл.
IF EXIST Y:\ GOTO EXIT
IF EXIST G:\flashd2.let SUBST Y: G:\
```

```
IF EXIST F:\flashd2.let SUBST Y: F:\
REM выход из командного файла
:EXIT
```

При выполнении измененного таким образом командного файла, сообщение об ошибке при выполнении SUBST исчезнет.

Одним из важнейших приемов при написании сложных командных файлов является анализ успешности выполнения конкретной команды или программы. Признаки ошибок при выполнении команд можно отслеживать, анализируя специальную переменную ERRORLEVEL, значение которой формируется при выполнении большинства программ. Обычно ERRORLEVEL равно нулю, если программа завершилась без ошибок и единице - при возникновении ошибки. Могут быть и другие значения, если они предусмотрены в выполняемой программе.

В качестве команды в строке командного файла можно использовать также командный файл. Причем, для передачи с возвратом обратно к точке выполнения вызывающего командного файла используется команда CALL. Попробуйте создать командный файл test.bat, следующего содержания:

```
@ECHO OFF
ECHO Вызов 1.bat
CALL 1.bat
ECHO Возврат.
```

В этом же каталоге, создайте второй файл под именем 1.bat, содержащий команду PAUSE, приостанавливающую выполнение командного файла до нажатия любой клавиши.

```
@ECHO OFF
Pause
```

При выполнении командного файла test.bat будет выдано на экран сообщение Вызов 1.bat и управление получит командный файл 1.bat с одной единственной командой pause. После нажатия клавиши на клавиатуре управление будет возвращено вызвавшему командному файлу на строку "ECHO Возврат." и на экран будет выдано

```
Возврат.
```

Если же в файле test.bat убрать CALL, оставив "1.bat", то возврат выполняться не будет.

Вызываемый командный файл может создавать переменные и присваивать им определенные значения, которые будут доступны для обработки в вызывающем файле. Попробуйте изменить файл test.bat на следующее содержимое:

```
@ECHO OFF
```

ECHO Вызов 1.bat
CALL 1.bat
ECHO Получено из файла %MYFILE% значение MYNUMBER=%MYNUMBER%,
а в файле 1.bat на следующее
@ECHO OFF
SET MYFILE="Very good 1.bat"
SET MYNUMBER=99

Используя передачу управления командному файлу, можно организовать его зацикливание. Добавите в конец файла test.bat строку:

CALL test.bat

Выйти из зацикливания командного файла можно по нажатию комбинации CTRL-Break. Возможно использование команды CALL для вызова процедуры внутри командного файла. В этом случае в качестве аргумента используется не имя внешнего файла, а метка:
call :proc1
....
:proc1
....
exit
....

Примеры командных файлов

Использование утилит командной строки и командных файлов нередко позволяют решить многие проблемы связанные с повседневной эксплуатацией компьютерной техники. Большинство системных администраторов и грамотных пользователей продолжают ими пользоваться, несмотря на то, что в Windows проявилось новое, более мощное и современное средство управления системой - WMI (Windows Management Instrumentation). Очевидно, не в последнюю очередь, это обусловлено простотой реализации и, тем не менее, - достаточной эффективностью использования командных файлов. Ниже приведены простые примеры с комментариями, которые демонстрируют некоторые возможности и способы применения .cmd и .bat .

Своя команда для создания новых файлов

В составе операционной системы Windows нет специальной команды для создания нового файла, но без нее можно легко обойтись несколькими способами:

Копирование с клавиатуры в файл

COPY CON myfile.txt

При выполнении этой команды данные с клавиатуры (стандартное устройство CON - консоль) будут заноситься в файл myfile.txt. Нажатие клавиши F6 или комбинации CTRL-Z завершит вывод.

Перенаправление вывода

ECHO 1 > myfile.txt

При выполнении этой команды будет создан файл myfile.txt, содержащий символ "1".

Комбинация перенаправления ввода и перенаправления вывода:

COPY CON > myfile.txt < xyz

При выполнении этой команды, как и в первом случае, используется копирование с консоли в файл, но вместо ручного ввода данных с клавиатуры используется ввод с несуществующего файла xyz. Система выдаст сообщение, о том, что такого устройства или файла не существует, но пустой файл myfile.txt будет успешно создан.

Еще проще использовать команду копирования из фиктивного устройства nul в файл. Использование устройства nul позволяет обойти стандартные операции ввода-вывода, которые для него реально не выполняются:

COPY NUL myfile.txt

При работе в командной строке часто приходится создавать новые пустые файлы, поэтому, стоит подготовить свой командный файл (например, с именем nf.bat), а имя нового создаваемого файла передавать ему в качестве параметра при запуске.

Содержимое файла:

```
@echo off
REM Создание пустого файла, имя которого задано в строке запуска
if "%1" EQU "" goto error
copy nul %1
goto exit
:error
ECHO ОШИБКА: Необходимо задать имя нового файла!
:exit
```

Для простоты использования, поместите этот командный файл в системный каталог (например, в C:\windows\system32) или любой другой, существующий в путях поиска, задаваемых значением переменной PATH). Теперь, в командной строке, находясь в любом каталоге можно одной командой создавать пустые файлы.

Командная строка:

nf.bat myfile.txt - создать файл с именем myfile.txt в текущем каталоге.

nf.bat C:\myfile.txt - создать файл в корневом каталоге диска C:

nf.bat "%USERPROFILE%\myfile.txt" - создать файл в каталоге профиля текущего пользователя.

Расширение командного файла (.bat) можно не набирать и команда еще больше упрощается:

nf myfile.txt

В тексте командного файла присутствует проверка, задано ли имя создаваемого файла в командной строке (if "%1%" EQU "" goto error), и если не задано - выводится сообщение об ошибке и командный файл завершает свою работу. Добавите в этот командный файл проверку на существование файла с именем, указанным в командной строке.

Присвоение съемному диску одной и той же буквы

Задача заключается в том, чтобы съемный USB диск (флэш диск) был доступен всегда под одной и той же буквой, независимо от того, на каком компьютере он используется и как он подключен. Для ее решения воспользуемся уже упоминаемой выше командой SUBST, но реализуем присвоение новой буквы диску с использованием подстановочного значения переменной %0, создаваемой системой при каждом запуске командного файла.

Выберем для съемного диска желаемую букву, например - X.

Некоторые из переменных окружения, в том числе и переменная %0 принимающая значение пути и имени выполняющегося командного файла, позволяют при определенной модификации с использованием специального признака - символа " ~ " получить ее частичное значение (расширение переменной). Например, не полный путь файла, а только его имя, или каталог расположения, или букву диска, с которого он был запущен или еще около десятка различных элементов, связанных с подстановочными значениями переменной %0.

Имя диска, с которого был запущен командный файл доступно как переменная %~d0.

Теперь создаем командный файл следующего содержания:

```
@echo off
subst X: %~d0\
```

что означает - создать виртуальный диск X:, которому сопо-

ставлен логический диск, являющийся частью пути данного командного файла. Если такой файл записать на флэшку, и выбрать присваиваемую букву диска поближе к концу алфавита (чтобы не оказалась занятой другим реальным дисковым устройством) то после его запуска, в системе будет создаваться новый диск всегда под одной и той же буквой.

Дополнительное представление о подстановочных значениях переменной %0 можно получить из командного файла следующего содержания:

```
@echo off
ECHO ОБРАБАТЫВАЕТСЯ ФАЙЛ - %0
ECHO Дата/время создания/изменения командного файла - %~t0
ECHO Путь командного файла - "%~f0"
ECHO Диск командного файла - %~d0
ECHO Каталог командного файла - "%~p0"
ECHO Имя командного файла - %~n0
ECHO Расширение командного файла - %~x0
ECHO Короткое имя и расширение - %~s0
ECHO Атрибуты командного файла - %~a0
ECHO Размер командного файла - %~z0
```

Создание архива, имя которого содержит дату и время

Решим следующую задачу - нужно создать архив файлов, находящихся в каталоге C:\Program Files\FAR. Имя архивного файла должно состоять из текущего времени (часы.минуты.секунды - ЧЧ.ММ.СС.rar), и помещен он должен в новый каталог, имя которого должно состоять из текущей даты (день.месяц.год - ДД.ММ.ГГГГ). Для архивирования будем использовать архиватор RAR. Формат запуска для создания архива:

RAR a -r < путь и имя архива > < Путь и имя архивируемых данных >

a - команда создания архива.

-r - ключ, определяющий архивирование подкаталогов (т.к. в исходной папке есть подкаталоги).

Таким образом, для решения задачи нужно правильно создать имена и пути для RAR. Для чего воспользуемся следующими исходными данными:

- в командных файлах можно получить доступ к текущей дате и текущему времени - переменные %DATE% и %TIME%;
- в командных файлах можно создавать временные переменные с помощью команды SET;

Значение временных переменных может быть сформировано на основе %DATE% и %TIME% путем пропуска и (или) замещения их частей с помощью специальной конструкции с использованием символа ~ и числового значения, определяющего группу символов из данных текущего значения переменной.

Дата, получаемая из переменной %DATE% при стандартных настройках региональных установок Windows 2000 выглядит следующим образом:

Пн 21.01.2014 - День недели (2 символа)-Пробел(1 символ)-дата(10 символов) - всего 13 символов. В Windows XP/Vista/7 день недели отсутствует, что несколько упрощает структуру даты. Для создания нового каталога в командной строке используется команда MD имя каталога.

Имя каталога нужно получить из текущей даты. Создаем в памяти временную переменную VDATE и присваиваем ей значение переменной окружения DATE, без первых 3-х символов (Пн и пробел) - 20.01.2014:

set VDATE=%date:~3%

В версиях Windows, где в значении принимаемом переменной DATE, отсутствует день недели (3 символа - "Пн "), значение VDATE получится не тем, что требуется. Чтобы не анализировать признаки наличия данного кода, можно воспользоваться и другим вариантом - не пропустить первые 3 символа (~3) от начала строки переменной DATE, а взять 10 символов от конца строки, указав число 10 со знаком "минус" - будет тот же результат - 20.01.2010

set VDATE=%date:~-10%

Создаем каталог на диске C:, имя которого = текущая дата из переменной VDATE:

MD C:\%VDATE%

После выполнения этой команды на диске C: будет создан каталог с именем 20.01.2014.

Можно обойтись без лишних операторов, связанных с формированием значения переменной VDATE, которую я использовал для упрощения понимания структуры создаваемого имени каталога:

MD %DATE:~-10% - создать каталог, имя которого будет представлено в виде текущей даты ДД.ММ.ГГГГ

Время, получаемое из переменной %TIME%, выглядит так: 14:30:59.93 - Часы, минуты, секунды, сотые доли секунды.

Сотые доли - это в имени файла архива, пожалуй, лишнее. Создаем временную переменную VTIME и присваиваем ей теку-

щее время без последних 3-х символов, т.е. пропускаем 0 символов от начала и отсекаем 3 символа от конца. Количество пропущенных и отсекаемых символов разделяются запятой:

```
set VTIME=%time:~0,-3%
```

Теперь VTIME = 14:30:59, но знак двоеточия в имени файла использовать нельзя, это специальный символ, использующийся в именах устройств (диск C:\). Поэтому, его придется заменить его на любой другой символ, допустимый в имени файла, например, точку. Для замены символов используется знак " = "

```
set VTIME=%VTIME::=.%
```

- заменить в переменной VTIME символ двоеточия на символ точки.

Переменная VTIME примет значение 14.30.59

Запустим архиватор:

```
rar.exe a -r C:\%VDATE%\%VTIME%.rar "C:\Program files\far\*.*"
```

Теперь можно создать командный файл с содержимым:

```
set VDATE=%date:~-10%
md c:\%VDATE%
set VTIME=%time:~0,-3%
set VTIME=%VTIME::=.%
rar.exe a -r C:\%VDATE%\%VTIME%.rar "C:\Program files\far\*.*"
```

Такой командный файл можно выполнять через автозагрузку, или как часть скрипта, при входе пользователя в домен, либо с помощью планировщика в заданное время, и у вас всегда будут в наличии упорядоченные по времени архивы критических данных.

Создания архива каталога "Мои Документы"

Этот командный файл создает архивы содержимого папки "Мои Документы" пользователей Win2K/XP, размещая их в каталоги

C:\ARHIV\Мои документы\Имя пользователя\Дата\время. При этом используются переменные окружения USERPROFILE, USERNAME, WINDIR. В файле используется команда rem, дающая некоторые комментарии:

```
@echo off
rem Задается переменная FROM - откуда брать данные для архивирования
set FROM=%USERPROFILE%\Мои Документы
rem Задается переменная TO - куда помещать архивы
```

```
set TO=C:\arhiv\Мои документы\%USERNAME%
rem Создадим каталог TO
md "%TO%\"
rem Сформируем имя подкаталога из текущей даты
set VDATE=%date:~-10%
rem Сформируем имя файла архива из текущего времени - 12:00:00.99
rem отбросим сотые доли секунды и заменим символ : на символ . Результат - 12.00.00
set vtime=%TIME:~0,-3%
set vtime=%vtime::=.%
rem Создадим подкаталог для файла архива
md "%TO%\%VDATE%"
rem Команда для архивирования. Ключ -r нужен для архивирования с вложенными папками
rem вариант для архиватора ARJ : arj.exe a -r "%TO%\%VDATE%\%VTIME%.arj" "%FROM%\*.*"
rem При использовании архиватора RAR:
rar.exe a -r "%TO%\%VDATE%\%VTIME%.rar" "%FROM%\*.*"
```

Если возникают проблемы связанные с неверной кодировкой символов русского алфавита в именах файлов и каталогов, воспользуйтесь командой CHCP для смены кодовой страницы

chcp 866 - установить кодовую страницу 866 (DOS-кодировка);

chcp 1251 - установить кодовую страницу 1251 (Windows-кодировка).

Этот командный файл можно значительно сократить, убрав ненужные переменные VTIME и VDATE, которые в данном примере, используются лишь для того, чтобы скрипт имел более наглядный и простой для понимания вид.

В операционных системах Windows XP/Vista/7 формат даты по умолчанию не содержит название дня недели. Если есть необходимость получить это значение без изменения настроек системы и использования дополнительного программного обеспечения, можно воспользоваться сценарием Hindows Script Host (WSH).

- создаем файл сценария для получения названия дня недели, пусть с именем weekday.vbs, и содержащим строку вывода на экран результата выполнения функции WeekDayName

WScript.Echo WeekDayName(Weekday(Now), True);

- выполняем скрипт WSH с использованием консольной версии программы обработки сценариев cscript.exe и подавлением

лишних сообщений (ключ //nologo)

cscript //nologo weekday.vbs

Пример командного файла для получения названия дня недели с использованием функции WeekDayName :

```
ECHO OFF
echo WScript.Echo WeekDayName(Weekday(Now), True) > weekday.vbs
for /f "Tokens=1*" %%i in ('cscript /nologo weekday.vbs') DO set DayName=%%i
echo %DayName%
REM Дальше можно использовать переменную DayName, а файл weekday.vbs – удалить
REM ERASE dayname.vbs
REM ...
```

Выполнение команд по расписанию

В операционных системах WINDOWS 2000/XP и старше существует утилита командной строки AT.EXE, позволяющая управлять задачами для планировщика заданий Windows, и таким образом, выполнить команду или пакетный файл в указанное время на локальном или удаленном компьютере. Естественно, для успешного функционирования команды AT необходимо, чтобы была запущена системная служба Планировщик заданий (обычно она существует и запускается автоматически при стандартной установке системы).

Примеры команды:

AT [\\имя_компьютера] [[код] [/DELETE] | /DELETE [/YES]]

AT [\\имя_компьютера] время [/INTERACTIVE] [/EVERY:день[,...] | /NEXT:день[,...]] "команда" где:

\\имя_компьютера - имя удаленного компьютера. Если этот параметр опущен, задача относится к локальному компьютеру;

код - порядковый номер запланированной задачи. Указывается если нужно отменить уже запланированную задачу с помощью ключа /delete;

/delete - отменить запланированную задачу. Если код задачи опущен, отменяются все задачи, запланированные для указанного компьютера;

/yes - не будет запроса на подтверждение при отмене всех запланированных задач;

время - время запуска команды;

/interactive - интерактивный режим, разрешение взаимодей-

ствия задачи с пользователем. Задачи, запущенные без этого ключа невидимы для пользователя компьютера; /every:день[,...] Запуск задачи осуществляется по указанным дням недели или месяца. Если дата опущена, используется текущий день месяца;

/next:день [,...] Задача будет запущена в следующий указанный день недели (например в следующий четверг). Если дата опущена, используется текущий день месяца; "команда" - Команда или имя командного файла.

Примеры использования:

Просмотр списка запланированных задач: AT;

Удаление уже спланированных задач: AT 3 /DELETE - удаление задачи с номером 3;

AT /DELETE /YES - удаление всех задач без запроса подтверждения;

Создание интерактивных задач

AT \\SERVER 15:21 /interactive notepad.exe - на компьютере SERVER в 15:21 запустить видимое для пользователя приложение "Блокнот" (notepad.exe)$ AT 15:30 /interactive regedit.exe - в 15:30 запустить видимый редактор реестра на своем компьютере.

Аналог "будильника" - всплывающие окна с текстом, напоминающие о необходимости каких-либо действий. Для посылки сообщения удаленному пользователю используется утилита NET.EXE в режиме отправки сообщения SEND. На компьютерах должна быть запущена служба сообщений, иначе NET SEND не будет работать:

AT 17:30 net.exe send COMP Пора домой - в 17:30 отправить сообщение "Пора домой" пользователю компьютера COMP;

AT \\PROXY 15:30 net.exe send COMP2 Test Message - создать задание на компьютере PROXY, чтобы в 15:30 им было отправлено сообщение "Test Message" на компьютер COMP2; AT 15:45 net.exe send имя своего компьютера Task Scheduler test - в 15:45 на своем компьютере показать сообщение "Task Scheduler test".

Для доступа к удаленному компьютеру и создания заданий, пользователь, выполняющий команду AT должен обладать соответствующими правами по отношению к удаленной системе.

Создаваемые командой AT задачи доступны для обработки в среде пользователя с помощью оснастки "Назначенные задания" Windows: Пуск -> Панель управления -> Назначенные задания - здесь можно просматривать, изменять и удалять созданные

командой AT задачи.

Остановка и запуск системных служб

Для остановки и запуска служб из командной строки, в любой версии Windows, можно воспользоваться командой NET.EXE:

NET.EXE STOP < имя службы >

NET.EXE START < имя службы >

В качестве параметра команды можно использование как короткое, так и полное имя службы ("Dnscache" - короткое, "DNS-клиент" - полное имя службы). Имя службы, содержащее пробелы заключается в двойные кавычки. Пример перезапуска службы "DNS-клиент":

net stop "DNS-клиент"

net start "DNS-клиент"

То же, с использованием короткого имени:

net stop Dnscache

net start Dnscache

Полное имя службы можно скопировать из: "Панель управления" -> "Администрирование" -> "Службы" -> Имя службы ->"Свойства" -> "Выводимое имя". То же самое, но в режиме командной строки: "Пуск" - "Выполнить" -services.msc.

Для управления службами гораздо удобнее воспользоваться утилитой PsService.exe из утилит PsTools. Утилита не требует установки и работает в любой OS Windows. Кроме запуска и остановки, позволяет выполнить поиск конкретной службы на компьютерах локальной сети, опросить состояние и конфигурацию службы, изменить тип запуска, приостановить службу, продолжить, перезапустить. Для работы с системными службами в Windows XP и старше, можно использовать утилиту sc.exe, позволяющую не только остановить/запустить службу, но и опросить ее состояние, параметры запуска и функционирования, изменить конфигурацию, а также работать не только с системными службами, но и с драйверами. При наличии соответствующих прав, можно управлять службами не только на локальной, но и на удаленной машине. Примеры:
sc.exe stop DNSCache - остановить службу DNSCache на локальном компьютере;

sc \\192.168.0.1 query DNSCache - опросить состояние службы DNSCache на компьютере с IP-адресом 192.168.0.1;

sc \\COMP start DNSCache запустить службу DNSCache на компьютере COMP
Подсказку по работе с утилитой можно получить, введя:

sc /?

Диалог с пользователем

Для диалога с пользователем можно использовать команду:

SET /P имя переменной = текст,

при выполнении которой, на экран выдается текстовое сообщение < текст > и ожидается ввод ответа. Пример - выполним запрос пароля и присвоим его значение переменной "pset":

```
set /p pset="Enter password - "
echo Password is - %pset%
```

Недостатком данного способа является невозможность продолжения выполнения командного файла при отсутствии ответа пользователя, поэтому очень часто вместо set используются сторонние программы. В составе операционных систем семейства Microsoft Windows имеется утилита командной строки CHOICE позволяющая довольно просто реализовать диалог с пользователем и проанализировать введенные им данные, однако в разных версия ОС утилита может присутствовать в стандартной поставке (Windows 7) или входить в наборы дополнительных программных инструментов (Resource Kit Windows XP). Простейшая версия - CHOICE.COM, работающая во всех ОС семейства Windows.

CHOICE выдает пользователю текстовое сообщение и ожидает выбора одного из заданных вариантов ответа (нажатия клавиш на клавиатуре). По результатам выбора формируется переменная ERRORLEVEL, значение которой равно порядковому номеру выбора. По умолчанию вариантов выбора два - Y или N. Если ответ равен Y - то ERRORLEVEL=1, если N - то ERRORLEVEL=2. Можно использовать более 2-х вариантов выбора и есть возможность задать выбор по умолчанию, когда пользователь за определенное время не нажал ни одной клавиши. Формат командной строки:

CHOICE [/C[:]choices] [/N] [/S] [/T[:]c,nn] [text]

/C[:]choices - определяет допустимые варианты выбора. Если не задано – Y/N

/N - не выдавать варианты выбора;

/S - строчные и заглавные буквы отличаются.

/T[:]c,nn - Выбор по умолчанию равен "c" через "nn" секунд

text - Строка текста выводимая в качестве запроса

Создадим командный файл, демонстрирующий использование CHOICE. Он будет реагировать на нажатие клавиш "1","2",3" и "0" . При нажатии "0" выполняется завершение, а при нажатии остальных - сообщение пользователю. Если в течение 10 секунд

ничего не нажато – завершение:

```
@ECHO OFF
:CHOICE
CHOICE /C:1230 /T:0,10 Ваш вариант
IF %ERRORLEVEL% EQU 4 GOTO EXIT
echo Ваш выбор=%ERRORLEVEL%
GOTO CHOICE
:EXIT
```

Теперь, используя CHOICE можно создавать командные файлы, логика работы которых может определяться пользователем.

Определение доступности IP-адреса

Для проверки доступности сетевого узла используется стандартная утилита ping.exe. Утилита выполняет отправку ICMP-пакета на проверяемый узел (эхо-запрос) и ожидает ответный пакет (эхо-ответ). Результат проверки никак не отражается в переменной ERRORLEVEL и может быть получен только в данных стандартного вывода ping. Ненулевое значение ERRORLEVEL утилита ping.exe формирует только в том случае, если заданы ошибочные параметры командной строки. Иными словами, в некоторых случаях, нужный результат выполнения определенной команды нельзя определить по значению переменной ERRORLEVEL, и приходится анализировать, например, результат текстового вывода.

Если внимательно посмотреть на сообщения программы ping.exe при опросе доступного и недоступного узла, то можно заметить, что они значительно отличаются:

ping 456.0.0.1 - ping на несуществующий адрес.

Ответ на такую команду может отличаться от конкретной версии утилиты, и может быть приблизительно таким:

При проверке связи не удалось обнаружить узел 456.0.0.1. Проверьте имя узла и повторите попытку.

ping yandex.ru - ping на адрес узла yandex.ru.

Ответ на ping доступного узла:

Обмен пакетами с yandex.ru [87.250.250.11] по 32 байт:

Ответ от 87.250.250.11: число байт=32 время=10мс TTL=55

Таким образом, для решения задачи определения доступности узла в командном файле, достаточно проанализировать характерные слова в выводе ping.exe при успешном ответе. Наиболее характерно в данном случае наличие слова TTL. Оно никогда не встречается при возникновении ошибки и состоит всего лишь

из символов английского алфавита. Для поиска "TTL" в результатах ping.exe удобнее всего объединить ее выполнение в цепочку с командой поиска строки символовFIND.EXE (конвейер ping и find). Справку по использованию можно получить командой find /?

Поиск текстовой строки в одном или нескольких файлах:

FIND [/V] [/C] [/N] [/I] [/OFF[LINE]] "строка" [[диск:][путь]имя_файла[...]]

где:

/V - вывод всех строк, НЕ содержащих заданную строку;

/C - вывод только общего числа строк, содержащих заданную строку;

/N - вывод номеров отображаемых строк;

/OFF[LINE] - не пропускать файлы с установленным атрибутом "Автономный";

/I - поиск без учета регистра символов;

"строка"- искомая строка;

[диск:][путь]имя_файла - один или несколько файлов, в которых выполняется поиск.

Если путь не задан, поиск выполняется в тексте, введенном с клавиатуры либо переданном по конвейеру другой командой.

Как видно из справки, find.exe можно использовать для поиска нужной строки символов в тексте, переданном по конвейеру командой ping.exe. Если текст найден, значение переменной ERRORLEVEL будет равно 0.

```
ping -n 1 COMPUTER | find /I "TTL" > nul
if %ERRORLEVEL%==0 goto LIVE
ECHO computer не доступен
подпрограмма обработки недоступного состояния
...
Exit
:LIVE - начало подпрограммы обработки состояния доступности узла
...
```

В конвейер добавлена команда перенаправления стандартного вывода на фиктивное устройство nul, т.е. подавление вывода. Ключ -n 1 задает однократный опрос узла COMPUTER для ping.exe.

Определение текущей версии Windows

Для определения версии операционной системы в процессе выполнения командного файла, можно воспользоваться поиском определенных фрагментов текста в результатах выполнения ко-

манд, отображающих сведения о системе. Например, во всех операционных системах семейства Windows (и даже в DOS) существует специальная команда VER, предназначенная для отображения сведений о версии ОС. В результате выполнения команды, например, в среде Windows XP, отображается текст:

Microsoft Windows XP [Версия 5.1.2600]

В среде Windows 7, текст отличается:

Microsoft Windows [Version 6.1.7600]

Таким образом, результат выполнения команды VER в среде разных версий Windows, всегда содержит определенный текст, характерный только для данной ОС, и задача определения версии решается довольно просто:

```
@echo off
set curr_OS=
REM
ver | find /i "5.0"
if %errorlevel% == 0 set curr_OS=Windows 2000
REM
ver | find /i "5.1"
if %errorlevel% == 0 set curr_OS=Windows XP
REM
ver | find /i "5.2.3"
if %errorlevel% == 0 set curr_OS=Windows Server 2003
REM
ver|find /i "6.0"
if %errorlevel% == 0 set curr_OS=Windows Vista
REM
ver | find /i "6.1">nul
if %errorlevel% == 0 set curr_OS=Windows 7
REM
if "%curr_OS%"=="" set curr_OS=Unknown
echo Текущая версия ОС - %curr_OS%
```

Можно также воспользоваться более информативным выводом команды NET CONFIG WORKSTATION. При выполнении в среде Windows XP вывод команды представляет собой следующий текст:

```
Имя компьютера                  \\COMP1
Полное имя компьютера           COMP1.Mydomain
Имя пользователя                USER2
Активная рабочая станция на
      NetbiosSmb (000000000000)
      NetBT_Tcpip_{F53DEAF8-0AF5-4875-B565-
```

8ED55C594769} (000D87009D28)
Версия программы Windows 2002
Домен рабочей станции Mydomain
DNS-имя домена рабочей станции Mydomain
Домен входа Mydomain
Интервал ожидания открытия COM-порта (с) 0
Отсчет передачи COM-порта (байт) 16
Таймаут передачи COM-порта (мс) 250
Команда выполнена успешно.

Для среды Windows 7 результат выполнения команды выглядит так:

Имя компьютера \\COMP1
Полное имя компьютера COMP1.Mydomain
Имя пользователя user2
Активная рабочая станция на
NetBT_Tcpip_{F53DEAF8-0AF5-4875-B565-8ED55C594769} (000D87009D28)
Версия программы Windows 7 Professional
Домен рабочей станции Mydomain
Домен входа Mydomain
Интервал ожидания открытия COM-порта (с) 0
Отсчет передачи COM-порта (байт) 16
Таймаут передачи COM-порта (мс) 250
Команда выполнена успешно.

Строка Версия программы . . . тоже может быть использована для определения версии Windows, в среде которой выполняется командный файл. Кроме того, в результатах выполнения команды NET CONFIG WORKSTATION для серверных версий Windows всегда присутствует слово Server.

```
@echo off
set curr_OS=
REM
net config workstation | find /i "Windows 2000"
if %errorlevel% == 0 set curr_OS=Windows 2000
REM
net config workstation | find /i "Windows 2002"
if %errorlevel% == 0 set curr_OS=Windows XP
REM
net config workstation | find /i "Server 2003"
```

```
if %errorlevel% == 0 set curr_OS=Windows Server 2003
REM
net config workstation|find /i "Windows Vista"
if %errorlevel% == 0 set curr_OS=Windows Vista
REM
net config workstation | find /i "Windows 7">nul
if %errorlevel% == 0 set curr_OS=Windows 7
REM Плюс поиск по "Professional"
net config workstation | find /i "Версия программы" | find "Professional"
if errorlevel 0 if not errorlevel 1 set curr_OS=Windows 7 PRO
REM Если версия неизвестна:
if "%curr_OS%"=="" set curr_OS=Unknown
echo %curr_OS%
```

Выключение компьютеров по списку, созданному на основе сетевого окружения

Выключение производится утилитой PsShutdown.exe. Сначала создается файл со списком компьютеров на основе сетевого окружения, а затем выполняется их поочередное выключение, при условии, что компьютер не свой (иначе он может выключиться до окончания выполнения командного файла). Содержимое файла:

```
rem @echo off
REM Здесь нужно задать
REM имя домена или рабочей группы для которых строится список машин для выключения:
set MyDomain=имя домена
REM
REM Создадим текстовый файл comps.txt со списком компьютеров с помощью NET VIEW
net view /DOMAIN:%MyDomain% > comps.txt
REM
REM FOR /F "параметры" - использование данных из файла
REM eol=K - не использовать строки, начинающиеся с "K" - "Команда выполнена успешно"
REM skip=4 - пропустить первые 4 строки в файле
REM tokens=1 - брать для обработки 1-е слово в строке
FOR /F "eol=K skip=4 tokens=1 " %%i in (comps.txt) do (
REM Свой компьютер выключать не будем
REM Если имя компьютера не равно COMPUTERNAME – выключаем
```

```
IF /I %%i NEQ %COMPUTERNAME% psshutdown -k -t 0 %%i
)
```

Вам нужно только подредактировать строку set MyDomain=, указав имя домена и, при необходимости, добавить параметры -u -p для psshutdown.exe .

В реальной жизни из списка выключаемых компьютеров нужно исключить несколько штук, для чего удобно использовать команду FIND в цепочке с net.exe в скрипте формирования списка на основе сетевого окружения. Данная команда используется для поиска строк в текстовом файле по шаблону. Ключ /V используется для поиска строк, не совпадающих с шаблоном. Для выключения компьютеров, исключая server1...server4 удобно использовать такой вариант:

```
net view | find "\\" | find /v "сервер1" | find /v "сервер2" | find /v "сервер3" | find /v "сервер4" > comps.txt
FOR /F "tokens=1 " %%i in (comps.txt) do shutdown.exe -f -s -m %%i
```

Работа с дисками, файлами и каталогами

Задача - определить буквы дисков, присутствующих в системе и записать результат в файл с именем tstdsk.txt текущего каталога. Можно воспользоваться выполнением команды IF EXIST в цикле FOR для набора из букв латинского алфавита, т.е. для каждой буквы диска проверить наличие корневого каталога командой:

```
IF EXIST буква диска:\
```

Сначала создаем пустой файл:

```
copy nul tstdsk.txt
```

Это действие необязательно, если файла не существует, но в противном случае, результаты будут дописываться в конец файла, и если в нем уже был список дисков от предыдущего исполнения командного файла, то он удвоится. Командасору nul tstdsk.txt для существующего файла установит нулевой размер данных, т.е. сделает его пустым. Окончательно, командный файл будет выглядеть следующим образом:

```
copy nul tstdsk.txt
for %%i in (a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z) DO
(
if exist %%i:\ echo Disk %%i: exist >> tstdsk.lst
)
```

Для обработки файлов определенного типа, например лю-

бых с расширением .tmp используется маска - *.tmp . Так, для удаления всех файлов *.tmp из каталога C:\TEMP можно воспользоваться командой ERASE (или DEL)

ERASE C:\TEMP*.TMP

DEL /Q C:\TEMP*.TMP

В масках файлов и каталогов возможно использование частичных имен:

ERASE C:\TEMP\A*.TMP - удалить все файлы с расширением .TMP, имя которых начинается с символа "A";

DIR *u*.* - выдать список всех файлов и подкаталогов текущего каталога, в имени которых содержится символ "u";

DIR C:*t.* - выдать список всех файлов и каталогов в корне диска C:, имя которых заканчивается символом "t".

Задача - получить список всех каталогов с подкаталогами на логическом диске и записать результат в текстовый файл. Для рекурсивной обработки каталогов диска будем использовать команду FOR /R:

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

Ключ /R означает выполнение команды для каталога [диск:]путь. Если в команде путь не задан, то обработка выполняется для текущего каталога. Простой пример удаления файлов с расширением .tmp из каталога C:\TEMP:

FOR /R C:\temp\ %%i IN (*.tmp) DO del %%i

При выполнении команды возможно использование подстановочных значений переменной цикла для получения имен дисков, папок, файлов и их характеристик. Полный список возможных значений в случае использования переменной с именем i:

%%~i - из переменной %i удаляются обрамляющие кавычки (");

%%~fi - переменная %i расширяется до полного имени файла;

%%~di - из переменной %i выделяется только имя диска;

%%~pi - из переменной %i выделяется только путь к файлу;

%%~ni - из переменной %i выделяется только имя файла;

%%~xi - из переменной %i выделяется расширение имени файла;

%%~si - полученный путь содержит только короткие имена;

%%~ai - переменная %i принимает значение атрибутов

файла;

%%~ti - переменная %i принимает значение даты /времени файла;

%%~zi - переменная %i принимает значение размера файла.

Возможно объединение нескольких операторов:

%%~dpi - переменная %i заменяется только на имя диска и путь;

%%~nxi - переменная %i заменяется только на имя файла и его расширение;

%%~fsIi - переменная %i заменяется только на полный путь с краткими именами;

%%~ftzai - переменная %I заменяется на строку, выдаваемую командой DIR.

Значение переменной %%pi внутри цикла команды FOR /R будет последовательно принимать значения путей папок, начиная с заданного набора [диск:]путь.

Так же, как и в предыдущем примере, желательно обнулить файл с результатами возможного предыдущего запуска данного командного файла:

```
REM Обнулить / создать файл для хранения списка каталогов C:\dirlist.txt
copy nul C:\dirlist.txt
REM Занесем первой строкой в пустой файл что-то вроде заголовка списка
Echo *** Список папок на диске C: *** >> C:\dirlist.txt
REM Сделать текущим каталогом корневой каталог диска C:
cd c:\
REM Выполнить для корневого каталога и всех вложенных каталогов, команду ECHO с
выдачей значения переменной %%~pi
for /R %%i in (C) DO (
ECHO Папка "%%~pi"  >> C:\dirlist.txt
)
```

В результате выполнения этого командного файла в корне диска C: будет создан файл dirlist.txt, содержащий список каталогов диска.

Если в цикле команды FOR /R используются подстановочные значения переменной %%I, то в качестве набора (in) не стоит использовать символ точки.

Задача - найти на диске файлы с расширением .log и скопировать их в каталог на другом логическом диске - D:\MUSOR. Же-

лательно проверить наличие каталога D:\MUSOR и при необходимости, создать его командой md, а также удалить из него все файлы, если они существуют, командой del . Затем выполнить переход в корневой каталог диска C: и выполнить в цикле команды FOR поиск файлов по маске *.log во всех подкаталогах.

```
REM подготовить каталог D:\MUSOR
if not exist D:\MUSOR md D:\MUSOR
REM удалить без подтверждения ( /Q) все файлы из каталога
del /Q D:\MUSOR\*.*
REM перейти в корень диска C:
cd c:\
REM Выполнить проверку наличия файлов с расширением *.log и скопировать их в
REM D:\MUSOR
for /R %%i in (c) DO (
if exist "%%~dpi*.log" copy "%%~dpi*.log" "D:\MUSOR\*.*"
```

Практика использования FOR /R показала, что не стоит использовать в качестве набора для обработки символ "точка" (конструкция in (.)), поскольку при использовании подстановочных значений, можно получить возврат из текущего каталога на уровень выше. В данном примере в качестве набора in используется любой не служебный символ. Команду копирования (copy) можно заменить на команду перемещения файлов (MOVE), что приведет к удалению файлов источников после копирования в каталог D:\MUSOR. Пример с копированием файлов с расширением .log рассмотренный выше имеет некоторые существенные недостатки - не обрабатываются скрытые файлы и папки, и в конечном каталоге, куда копируются файлы (D:\MUSOR) не создаются подкаталоги с теми же именами, которые принадлежат путям исходных копируемых файлов. Для устранения этих недостатком можно использовать немного другой скрипт :

```
@echo off
REM подготовить каталог D:\MUSOR - удалить его и его подкаталоги командой RD
RD /S /Q D:\MUSOR
REM Создадим каталог заново
MD D:\MUSOR
REM Задаем начальную папку для обработки в команде FOR - C:\
for /R C:\ %%i in (C) DO (
xcopy "%%~dpi*.log" "D:\MUSOR%%~pi*.*" /H /R /Q /Y
```

)

Для копирования используется команда xcopy с ключами:

/H - копировать скрытые файлы.

/R - разрешение на замену файлов с атрибутом "Только чтение"

/Q - не отображать имена копируемых файлов

/Y - разрешать перезаписывать существующие файлы.

Подсказку по использованию команды XCOPY можно получить при вводе:

help xcopy

xcopy /?

При обработке строки xcopy "%%~dpi*.log" "D:\MUSOR%%~pi*.*" /H /R /Q /Y в цикле FOR, в качестве источника копирования будет выбираться C:\текущий путь*.log а в качестве приемника - D:\MUSOR\текущий путь\имя копируемого файла. Похожий подход можно использовать для обнаружения и копирования исполняемых файлов (*.exe) из каталога временных файлов, задаваемого переменной TEMP. Бывает полезно для поиска вредоносных программ.

REM @echo off

REM подготовить каталог D:\MUSOR - удалить командой RD

RD /S /Q D:\MUSOR

REM Создадим каталог заново

MD D:\MUSOR

REM Задаем начальную папку для обработки (%TEMP%) и выполняем FOR

for /R "%TEMP%" %%i in (C) DO (

xcopy "%%~dpi*.exe" "D:\MUSOR%%~pi*.*" /H /R /Q

)

При работе с содержимым каталогов удобно использовать команды запоминания текущего каталога и перехода в новыйPUSHD и команды восстановления ранее запомненного текущего каталога POPD:

PUSHD "%TEMP%"

Echo Работаем в каталоге временных файлов

REM новый каталог стал текущим и можно использовать относительные пути

REM Выдать список exe-файлов текущего каталога (%TEMP%) командой DIR

DIR *.exe

REM Восстановить путь, запомненный командой PUSHD

POPD

)

Echo Вернулись в исходный каталог.

Работа с графическими приложениями Windows

Допустим, вам нужно из одного и того же командного файла запустить notepad.exe и cmd.exe. Если просто вставить строки

notepad.exe

cmd.exe,

то после запуска notepad.exe выполнение командного файла приостановится и пока не будет завершен notepad, cmd.exe не запустится. Самый простой способ обойти эту проблему - использовать стандартную команду Windows start. Полную справку по использованию можно получить по:

start /?

Создайте командный файл следующего содержания:

start /MAX notepad.exe

start "This is CMD.EXE" /MIN cmd.exe

net send %COMPUTERNAME% NOTEPAD and CMD running.

После выполнения этого командного файла вы увидите стартовавшие, в развернутом окне (ключ /MAX) блокнот, в свернутом окне (ключ /MIN) командный процессор CMD.EXE и окно с сообщением net.exe. Стандартный заголовок окна cmd.exe заменен на текст "This is CMD.EXE". Обратите внимание на то что заголовок окна можно опускать, но особенность обработки входных параметров командой start может привести к неожиданным результатам при попытке запуска программы, имя или путь которой содержит пробел(ы). Например при попытке выполнить следующую команду:

start "C:\Program Files\FAR\FAR.EXE"

Из-за наличия пробела в пути к исполняемому файлу, строка для запуска FAR.EXE должна быть заключена в двойные кавычки, однако формат входных параметров для start предполагает наличие заголовка окна, также заключаемого в двойные кавычки, в результате чего "C:\Program Files\FAR\FAR.EXE" интерпретируется не как исполняемая программа, а как заголовок окна. Для того, чтобы подобного не случилось нужно использовать любой, пусть даже пустой, заголовок:

start "" "C:\Program Files\FAR\FAR.EXE"

Если вам все же потребуется расширенное управление окнами приложений, придется воспользоваться сторонним программным обеспечением, например, CMDOW.

Из-за специфического поведения эта утилита большинством

антивирусов определяется как вирус, поэтому для нормальной работы нужно занести ее в исключения антивируса.

Cmdow.exe - крошечная утилита, работающая в Windows NT4/2000/XP/2003 без установки. Позволяет получить список окон, перемещать, изменять размеры, переименовывать, сворачивать/разворачивать, активировать/деактивировать, закрывать, скрывать окна приложений и многое другое. Справку можно получить по команде:

cmdow /?

Используется около 30 ключей. Некоторые примеры:

Получение информации об окнах:

cmdow.exe или cmdow.exe > wins.txt - выдать информацию обо всех окнах на экран или в файл wins.txt

cmdow /T - выдать информацию об окнах, отображаемых на панели задач рабочего стола.

Информация содержит колонки:

Handle - дескриптор окна - шестнадцатеричное число, связанное с данным окном.

Lev - уровень окна. Приложение может быть многооконным с несколькими уровнями окон.

Pid - идентификатор процесса, породившего окно.

-Window status - состояние окна (видимое - Vis, скрытое - Hid, активное - Act, свернутое - Min и т.п.

Image - программа вызвавшая окно.

Caption - название окна

Манипулировать окнами можно используя название окна, или его дескриптор. Если название окна содержит пробелы, то оно заключается в двойные кавычки. Если имеются русские буквы, то должна использоваться DOS-кодировка. Символ@ используется для указания текущего окна. Иногда проще использовать дескриптор окна, а не его название. Полезным может быть и использование команды поиска по строке find.exe, выполняемой в цепочке с cmdow:

cmdow.exe | find.exe /I "hid" > wins.txt - в файл wins.txt попадут только строки содержащие шаблон "hid" и мы получим список скрытых окон.

cmdow.exe | find.exe /I "MyIE" > wins.txt - список окон приложения MyIE.

Манипулирование окнами

Если вы хотите, чтобы ваш командный файл выполнялся скрытно, добавьте в него строку:

cmdow @ /HID - скрыть текущее окно.

Ниже командный файл с комментариями, демонстрирующий возможности работы cmdow:

```
@ECHO OFF
REM Свернуть все окна - /MA
cmdow /MA
REM запустить cmd.exe с заголовком окна MyCMD
start "MyCMD" cmd.exe
REM ждать 5 секунд
call :wait5s
REM
:M1
REM Скрыть окно MyCND
cmdow MyCMD /hid
call :wait5s
REM Сделать видимым
cmdow MyCMD /vis
call :wait5s
REM Переместить в верхний левый угол экрана и развернуть окно
cmdow MyCMD /MOV 0 0
cmdow Mycmd /max
call :wait5s
REM Изменить размер на 320 x 240 и переместить вправо на 320 точек
cmdow MyCMD /MOV 320 0 /SIZ 320 240
call :wait5s
REM Переместить окно в точку с координатами 320 x 240 и изменить размер на 350x50
cmdow MYCMD /MOV 320 240 /SIZ 350 50
call :wait5s
REM Восстановить окно
cmdow MYCMD /RES
call :wait5s
REM Восстановить и сделать активным окно этого командного файла
cmdow @ /RES /ACT
ECHO Для завершения нажмите CTRL-C (CTRL-Break)
call :wait5s
call :wait5s
REM Зацикливание - переход к метке :M1
GOTO M1
```

```
REM Подпрограмма  задержки на 5секунд
:wait5s
@ping -n 5 localhost > nul
```

Пример командного файла, закрывающего окна Проводника Интернет (IEXPLORE.EXE):

```
@echo off
:M1
for /f "tokens=1-2,8"  %%a in ('cmdow') do (
if /i "%%c"=="IEXPLORE" if "%%b"=="1" cmdow %%a /END
> nul
)
goto M1
```

Работает это следующим образом. Из выходных данных CMDOW берется первое, второе и 8-е поля. Первое - дескриптор окна (Handle), второе - уровень (Lev), третье - имя программы (Image). В цикле выполняется cmdow и если в ее выводе имеется строка, где имя программы IEXPLORE и уровень окна 1 выполняется cmdow <дескриптор> /END. Пока этот командный файл выполняется, запустить "Обозреватель интернета" не получится. а если в начало командного файла добавить "cmdow @ /hid" - то будет скрыто и его окно.

Перекодировка текстовых файлов

В рассматриваемом примере нужно преобразовать исходный текстовый файл в DOS-кодировке в новый текстовый файл в Windows-кодировке. В качестве механизма перекодировки используется смена кодовой страницы командой CHCP и построчная выдача содержимого исходного файла командой ECHO с перенаправлением вывода в новый файл. Для DOS-кодировки используется кодовая страница 866, для Windows-кодировки - 1251. В примере исходный файл называется 866.txt, а файл с перекодированными данными - 1251.txt:

```
@echo off
chcp 866 >nul
for /f "tokens=*" %%i in (866.txt) do call:to1251 "%%i"
exit
:to1251
chcp 1251 >nul
echo %~1 >>1251.txt
chcp 866 >nul
exit /b
```

Аналогичный подход можно использовать и для преобразования текста из Windows - кодировки (кодовая страница 1251) в DOS-кодировку (кодовая страница 866). Естественно, такая перекодировка не может учитывать пустые строки и форматирование текста с помощью спецсимволов, поскольку команда ECHO не позволяет работать с такими форматами данных.

Своеобразным современным стандартом программы для перекодировки файлов считается, портированная из Unix утилита iconv (в составе библиотеки libiconv):

iconv [-c] [-s] [-f encoding] [-t encoding] [inputfile ...]

Входная кодировка задаётся ключом -f, а выходная - ключом -t . Если ключи не заданы, используется кодировка для языка системы по умолчанию. Все входные файлы читаются по очереди, если не задан параметр входного файла, то используется стандартный ввод, а конвертируемый текст выводится на стандартный вывод. Когда задана опция -c, символы, которые не могут быть преобразованы просто выбрасываются. В противном случае при появлении подобной ошибки программа аварийно завершается.

Когда задана опция -s, сообщения об ошибках не выводятся. Ключ -l позволяет получить список доступных кодировок. Утилита позволяет перекодировать текст, практически, из любой кодировки в любую.

Часто встречающиеся ошибки при написании командных файлов

Командный файл вручную выполняется успешно, но запущенный с помощью планировщика не работает.

Обычно, это вызвано тем, что вы не учитываете тот факт, что на момент выполнения вашего командного файла переменные среды могут быть совсем другими, чем на момент его написания и запуска из командной строки. Например, в командном файле используется запуск приложения myprog.exe, находящегося в каталоге SCRIPTS на диске D: . Если в командном файле используется имя модуля без полного пути MYPROG.EXE и если каталог D:\SCRIPTS не прописан в путях поиска (переменная PATH) то модуль MYPROG.EXE может быть найден и выполнен только если текущим каталогом является D:\SCRIPTS. Но если вы укажете полный путь к myprog.exe:

D:\SCRIPTS\myprog.exe,

то программа будет найдена и выполнена в любом случае. Кроме того, нередко программа, указанная в командном файле

использует для поиска своих компонент (dll, ini и т.п.) собственный каталог. Но на момент ее выполнения текущим каталогом может быть любой (чаще всего - системный каталог Windows). Естественно, компоненты не находятся и программа не выполняется. Для устранения проблемы добавьте в командный файл команды, обеспечивающие переход в нужный каталог. Например, программа myprog.exe должна выполняться в каталоге D:\SCRIPTS:

```
Rem Сменим текущий диск
D:
Rem перейдем в каталог SCRIPTS
CD D:\SCRIPTS
myprog.exe
```

Неправильно отображаются русские имена файлов, служб и т.п.

Причина в том, что при создании командных файлов вы использовали текстовый редактор, в котором русские символы представлены не в DOS-кодировке. Если в приведенном выше примере перезапуска службы "DNS-клиент" вы используете неверную кодировку, то русская часть имени службы не будет опознана из-за неверной кодировки и будет выдано сообщение, что указанная служба не установлена. Чтобы избежать проблем с русскими символами в командных файлах, используйте редактор с поддержкой DOS-кодировки, например, встроенный редактор файлового менеджера FAR. Переключение между кодировками в редакторе осуществляется нажатием F8 . с помощью FAR можно легко осуществлять перекодировку, скопировав (вырезав) текст в буфер обмена, затем нажав F8 и вставив текст из буфера.

Командный файл выполняется на одном компьютере успешно, но на другом - не работает.

Обычно это вызвано применением в командных файлах абсолютных значений вместо переменных среды окружения. Вместо C:\WINDOWS правильнее использовать %SYSTEMROOT%, потому, что на другом компьютере система может быть установлена в другой каталог или на другой диск. Старайтесь вместо имени командного файла использовать переменную %0 и ее подстановочные варианты (%~d0 - диск с которого запущен сценарий, %~dp0 - полный путь и т.д.). Строки с переменными, принимающими значения имен файлов и каталогов лучше заключать в кавычки. Командная строка DIR %ProgramFiles% не выдаст вам содержимого каталога C:\Program Files , поскольку из-за наличия пробела

будет интерпретирована как DIR C:\Program. Командная строка DIR "%ProgramFiles%" выполнится верно. Старайтесь использовать комманды Setlocal и Endlocal, чтобы не оставлять мусор из переменных, созданных или модифицированных командным файлом .<br.< span=""></br.<>

Использование командных файлов в сценариях регистрации пользователей

Командные файлы удобно использовать для выполнения каких-либо действий при регистрации пользователя в домене. Делается это с помощью вкладки Profile свойств пользователя домена.

Сами командные файлы должны находиться в сетевой папке Netlogon (WINDOWS\SYSVOL\DOMAIN\SCRIPTS) контроллера домена.

Порядок выполнения работы

Составить командный файл для загрузки системы в минимальной конфигурации:

1. Включить команду для того, чтобы обрабатываемые командным процессором строки не выдавались на экран.
2. Включить команду для просмотра и редактирования командного файла, содержащего символы русского алфавита с использованием редактор с стандартного приложения "Блокнот" (notepad.exe).
3. Задать цвет фона и цвет символа.
4. Вывести справку в файл с именем help.txt.
5. Ввести команду задания и модификикации пути поиска исполняемых программ в каталогеC:\Windows, и, если результат неуспешен, в C:\windows\system32; указать количество процессоров; архитектуру процессора; идентификатор процессора; уровень (номер модели) процессора; версию процессора; формат приглашения командной строки; букву системного диска; каталог ОС Windows.
6. Реализовать просмотр действующего значения какой-либо переменной.

Контрольные вопросы

1. Дать понятие командного файла.
2. Как запустить командный процессор в интерактивном режиме?

3. Какая кодовая страница используется при работе в среде Windows?

4. Какая команда выводит полную справку?

5. Что такое переменные окружения?

6. Как передавать параметры командной строки и использовать их значения в операциях внутри самого командного файла?

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 15. «ФАЙЛЫ ПАКЕТНОЙ ОБРАБОТКИ»

Теоретическая часть

Целью данной работы является:

- изучение назначения и основных возможностей командных файлов (Файлов пакетной обработки) операционных систем, построенных на платформе Windows NT;

- знакомство со специальными командами, используемыми в командных файлах;

- исследование стандартных потоков ввода-вывода и их перенаправление.

Язык командных файлов

Командный файл – это текстовый файл (в коде ASCII), состоящий из группы команд. Правила идентификации командных файлов совпадают с общими правилами идентификации файлов. Единственное исключение — командный файл всегда записывается на диск с расширением «.BAT» и/или «.CMD» (для операционных систем Windows на платформе NT).

Обратиться к командному файлу крайне просто. Набирается команда старта – имя файла, и нажимается клавиша Enter. После введения команды файл выбирается из рабочего каталога указанного или рабочего диска. Если в рабочем каталоге его нет, то поиск файла будет производиться в каталогах, описанных системной переменной %PATH%. При нахождении файла первая из его команд загружается в память, отображается на экране и выполняется. Этот процесс повторяется последовательно для всех команд файла (от первой до последней команды).

Выполнение командного файла можно прервать в любой момент, нажав на клавиши Ctrl-Break (Ctrl-C).

Организация командного файла. Существует несколько способов организации командных файлов. Файл можно создать с помощью любого текстового редактора или введением команд непосредственно с клавиатуры. В этом случае ввод оформляется файлом и записывается на диск.

В сеансе DOS клавиатура называется «CON» (CONsole) Для организации файла используется команда «COPY CON:». Наберите команду и имя создаваемого файла. Например, для создания файла «SAMPLE.BAT» введите:

C:\>COPY CON: SAMPLE.BAT

После этого введите составляющие файл команды. Набрав последнюю команду, одновременно нажмите клавиши Ctrl-Z (или функциональную клавишу F6) и клавишу Enter.

Стандартные потоки ввода-вывода и перенаправление потоков. Термин CONsole используется для обозначения стандартных потоков ввода-вывода. Когда говорят о вводе с консоли, подразумевается ввод с клавиатуры. Когда говорят о выводе на консоль, подразумевают вывод на экран монитора. Существуют специальные символы для перенаправления стандартных потоков ввода-вывода.

> приемник — перенаправить стандартный вывод в приемник (если файл-приемник существует, то он будет создан заново).

>> приемник — перенаправить стандартный вывод в приемник (если файл-приемник существует, то он будет сохранен, а информация будет записана в конец файла).

< источник — перенаправить стандартный ввод из источника.

передатчик | приемник — передает вывод одной команды на вход другой.

Замещаемые параметры. Внутри командного файла допускается использование замещаемых параметров. Параметр — это символьная переменная, расположенная в командной строке после имени команды. Он содержит дополнительную информацию, необходимую операционной системе при обработке команды. Параметром, напри- мер, может быть имя файла, к которому относится действие команды. Замещаемый параметр — это специальная переменная, которая в процессе выполнения команды подменяется обычным параметром (например, именем файла). В командном файле замещаемый параметр обозначается знаком процента % и цифрой от 0 до 9. Таким образом, командный файл может включать до десяти замещаемых параметров. Символьные переменные, предназначенные для подмены замещающего параметра, вводятся в командной строке при обращении к команд- 6 ному файлу — набирается команда старта (имя файла) и список параметров в порядке, соответствующем последовательности замещаемых параметров внутри файла

Параметры заменяются в порядке следования символьных переменных в командной строке. Первая переменная подменяет параметр %1, вторая – параметр %2 и т.д. Вместо замещаемого параметра %0 автоматически подставляется спецификация (имя) командного файла.

При введении замещаемых параметров командный файл

становится более гибким. Поясним это на примере. Предположим, что на диске имеется несколько файлов, которые нужно копировать после каждой корректировки. В рассмотренном выше примере командный файл использовался для копирования конкретного файла. Этим же ко- мандным файлом можно воспользоваться и для копирования любого файла. В этом случае вместо имени копируемого файла подставляется замещаемый параметр. Имя копируемого файла будет вводиться в командной строке при обращении к командному файлу.

Назовем наш командный файл «COPYALL.BAT». Введем в нем:

COPY %1 A:

При обращении к файлу набирается его имя и через пробел – имя копируемого файла (в нашем примере «SHOPLIST.DOC»). Введи- те команду:

C:\>COPYALL.BAT SHOPLIST.DOC

На экран выводится следующая команда:

C:\>COPY SHOPLIST.DOC A:

1 File(s) copied

DOS автоматически подставила имя файла на место замещаемо- го параметра %1. Усложним пример. Организуем командный файл «DIFNUM.BAT», автоматически копирующий любой указанный файл и присваивающий копии любое указанное имя:

COPY %1 A:%2

Для обращения к этому файлу наберите его имя, имя копируемого файла, в нашем примере «NEW.DOC», и имя копии «OLD.DOC»:

C:\>DIFNUM NEW.DOC OLD.DOC

На экране появляется следующая команда файла «DIFNUM.BAT»:

C:\>COPY NEW.DOC A:OLD.DOC

1 File(s) copied

Первое имя в командной строке «NEW.DOC» поставлено вместо замещаемого параметра %1. Второе имя «OLD.DOC» – вместо замещаемого параметра %2.

Замещаемые параметры и замещаемые символы. Параметр в командной строке команды старта командного файла может включать замещаемые символы «?» и «*». Если замещаемый символ вводится для обозначения группы параметров, то команда выполняется по количеству параметров в группе (т.е. один раз для каждого параметра). Рассмотрим командный файл:

COPY %1 CON:

Этот файл копирует на экран (CON) файл, описанный замещаемым параметром %1 (DISPLAY.BAT). Имя копируемого файла указывается в командной строке при обращении к командному файлу. Если указанный файл найден, его содержимое выводится на экран.

Этот файл копирует на экран (con) файл, описанный замещаемым параметром %1. Имя копируемого файла указывается в командной строке при обращении к командному файлу. Если указанный файл найден, его содержимое выводится на экран. Итак, командный файл «DISPLAY.BAT» записан на диск. Введем команду:

C:\>DISPLAY *.TXT

Все файлы рабочего диска с соответствующей спецификацией будут выведены на экран. Если имя копируемого файла включает обозначение процента, то при введении его в командную строку знак процента набирается два раза подряд. Например, имя «HIHO%.TXT» в командной строке должно быть представлено как «HIHO%%.TXT».

Некоторые команды DOS (Windows)

Для получения полного списка команд DOS, поддерживаемых вашей операционной системой Windows, построенной на платформе NT, необходимо ввести команду :

HELP

Вот ее возможный результат: Для получения сведений об определенной команде наберите HELP

Команда	Описание
ASSOC	Вывод либо изменение сопоставлений по расширениям имен файлов.
AT	Выполнение команд и запуск программ по расписанию.
ATTRIB	Отображение и изменение атрибутов файлов.
BREAK	Включение/выключение режима обработки комбинации клавиш CTRL+C.
CACLS	Отображение/редактирование списков управления доступом (ACL) к файлам. CALL — Вызов одного пакетного файла из другого.
CD	Вывод имени либо смена текущей папки.

CHCP	Вывод либо установка активной кодовой страницы.
CHDIR	Вывод имени либо смена текущей папки.
CHKDSK	Проверка диска и вывод статистики.
CHKNTFS	Отображение или изменение выполнения проверки диска во время загрузки. CLS — Очистка экрана.
CMD	Запуск еще одного интерпретатора командных строк Windows.
COLOR	Установка цвета текста и фона, используемых по умолчанию.
COMP	Сравнение содержимого двух файлов или двух наборов файлов.
COMPACT	Отображение/изменение сжатия файлов в разделах NTFS.
CONVERT	Преобразование дисковых томов FAT в NTFS. Нельзя выполнить преобразование текущего активного диска.
COPY	Копирование одного или нескольких файлов в другое место.
DATE	Вывод либо установка текущей даты.
DEL	Удаление одного или нескольких файлов.
DIR	Вывод списка файлов и подпапок из указанной папки.
DISKCOPY	Копирование содержимого одного гибко- го диска на другой.
DOSKEY	Редактирование и повторный вызов командных строк; создание макросов. ECHO — Вывод сообщений и переключение режима отображения команд на экране. ENDLOCAL— Конец локальных изменений среды для пакетного файла.
ERASE	Удаление одного или нескольких файлов.
DISKCOMP	Сравнение содержимого двух гибких дисков.

EXIT	Завершение работы программы CMD.EXE (интерпретатора командных строк). FC — Сравнение двух файлов или двух наборов файлов и вывод различий между ними. FIND — Поиск текстовой строки в одном или нескольких файлах.
FINDSTR	Поиск строк в файлах.
FOR	Запуск указанной команды для каждого из файлов в наборе.
FORMAT	Форматирование диска для работы с Windows. FTYPE — Вывод либо изменение типов файлов, используемых при сопоставлении по расширениям имен файлов. GOTO — Передача управления в отмеченную строку пакетного файла.
GRAFTABL	Позволяет Windows отображать расширенный набор символов в графическом режиме.
HELP	Выводит справочную информацию о командах Windows.
IF	Оператор условного выполнения команд в пакетном файле.
LABEL	Создание, изменение и удаление меток тома для дисков.
MD	Создание папки. MKDIR — Создание папки.
MODE	Конфигурирование системных устройств.
MORE	Последовательный вывод данных по частям размером в один экран.
\MOVE	Перемещение одного или нескольких файлов из одной папки в другую.
PATH	Вывод либо установка пути поиска исполняемых файлов.

PAUSE	Приостановка выполнения пакетного файла и вывод сообщения.
POPD	Восстановление предыдущего значения текущей активной папки, сохраненного с помощью команды PUSHD.
PRINT	Вывод на печать содержимого текстовых файлов.
PROMPT	Изменение приглашения в командной строке Windows.
PUSHD	Сохранение значения текущей активной папки и переход к другой папке.
RD	Удаление папки. RECOVER— Восстановление читаемой информации с плохого или поврежденного диска.
REM	Помещение комментариев в пакетные файлы и файл CONFIG.SYS.
REN	Переименование файлов и папок.
RENAME	Переименование файлов и папок.
REPLACE	Замещение файлов.
RMDIR	Удаление папки.
SET	Вывод, установка и удаление переменных среды Windows.
SETLOCAL	Начало локальных изменений среды для пакетного файла.
SHIFT	Изменение содержимого (сдвиг) подставляемых параметров для пакетного файла.
SORT	Сортировка ввода.
START	Запуск программы или команды в отдельном окне.
SUBST	Сопоставляет заданному пути имя диска.
TIME	Вывод и установка системного времени.
TITLE	Назначение заголовка окна для текущего сеанса интерпретатора командных строк CMD.EXE.

TREE	Графическое отображение структуры папок заданного диска или заданной папки.
TYPE	Вывод на экран содержимого текстовых файлов.
VER	Вывод сведений о версии Windows.
VERIFY	Установка режима проверки правильности записи файлов на диск.
VOL	Вывод метки и серийного номера тома для диска.
XCOPY	Копирование файлов и дерева папок.

Чтобы получить информацию о какой-либо команде операционной системы можно также в командной строке набрать имя команды и через пробел указать знак /?. Например,

C:\>PAUSE /?

Далее приводится основной синтаксис некоторых команд, необходимых для выполнения лабораторной работы.

ECHO

ECHO [ON | OFF] — переключение режима отображения команд на экране.

ECHO [сообщение] — вывод сообщений.

Введите ECHO без параметра для определения текущего значения этой команды.

Введите ECHO. (с точкой) для получение пустой строки. @ — знак экранирования. Отключает вывод на экран текущей строки.

GOTO — передача управления содержащей метку строке пакетного файла.

GOTO метка

метка — строка пакетного файла, оформленная как метка.

Метка должна находиться в отдельной строке и начинаться с двоеточия.

IF — оператор условного выполнения команд в пакетном файле.

IF [NOT] ERRORLEVEL число команда

IF [NOT] строка1==строка2 команда

IF [NOT] EXIST имя файла команда

NOT — обращает истинность условия: истинное условие становится ложным, а ложное — истинным.

ERRORLEVEL число — условие является истинным, если код возврата последней выполненной программы не меньше указанного числа.

строка1==строка2 — это условие является истинным, если указанные строки совпадают.

IF (%1)==() — проверка на пустой параметр.

EXIST имя файла — это условие является истинным, если файл с указанным именем существует.

команда — задает команду, выполняемую при истинности условия. За этой командой может следовать ключевое слово ELSE, служащее для указания команды, которая должна выполняться в том случае, если условие ложно.

Предложение ELSE должно располагаться в той же строке, что и команда, следующая за ключевым словом IF. Например:

```
IF EXIST имя_файла. (
del имя_файла.
)
ELSE (
echo имя_файла. missing.
)
```

Следующий пример содержит ОШИБКУ, поскольку команда del должна заканчиваться переходом на новую строку:

IF EXIST имя_файла. del имя_файла. ELSE echo имя_файла. missing

Следующий пример также содержит ОШИБКУ, поскольку команда ELSE должна располагаться в той же строке, что и команда, следующая за IF: IF EXIST имя_файла. del имя_файла. ELSE echo имя_файла. missing Вот правильный пример, где все команды расположены в одной строке:

```
IF EXIST имя_файла. (del имя_файла.)
ELSE echo имя_файла. missing
```

PAUSE — приостановка выполнения пакетного файла и вывод сообщения: Для продолжения нажмите любую клавишу . . .

DIR — вывод списка файлов и подкаталогов из указанного каталога.

DIR [диск:][путь][имя_файла] [/A[[:]атрибуты]] [/B] [/C] [/D] [/L] [/N] [/O[[:]порядок]] [/P] [/Q] [/S] [/T[[:]время]] [/W] [/X] [/4]

[диск:][путь][имя_файла]

Диск, каталог и/или файлы, которые следует включить в список. /A Вывод файлов с указанными атрибутами. атрибуты:

- D Каталоги
- R Доступные только для чтения
- H Скрытые файлы
- A Файлы для архивирования

- S Системные файлы

- Префикс «-» имеет значение НЕ

/B Вывод только имен файлов.

/C Применение разделителя групп разрядов для вывода размеров файлов (по умолчанию). Для отключения этого режима служит ключ /-C.

/D Вывод списка в несколько столбцов с сортировкой по столбцам.

/L Использование нижнего регистра для имен файлов.

/N Отображение имен файлов в крайнем правом столбце.

/O Сортировка списка отображаемых файлов. порядок: - N По имени (алфавитная) - S По размеру (сперва меньшие) - E По расширению (алфавитная) - D По дате (сперва более старые) - G Начать список с каталогов - Префикс «-» обращает порядок

/P Пауза после заполнения каждого экрана.

/Q Вывод сведений о владельце файла.

/S Вывод списка файлов из указанного каталога и его под-ката- логов.

/T Выбор поля времени для отображения и сортировки время: - C Создание - A Последнее использование - W Последнее изменение

/W Вывод списка в несколько столбцов.

/X Отображение коротких имен для файлов, чьи имена не соответствуют стандарту 8.3. Формат аналогичен выводу с ключом

/N, но короткие имена файлов выводятся слева от длинных. Если короткого имени у файла нет, вместо него выводятся пробелы.

/4 Вывод номера года в четырехзначном формате Стандартный набор ключей можно записать в переменную среды DIRCMD. Для отмены их действия введите в команде те же ключи с префиксом «-», например:

/-W. 15 MD — создание каталога. MKDIR [диск:]путь MD [диск:]путь CD — вывод имени либо смена текущего каталога. CHDIR [/D] [диск:][путь] CHDIR [..] CD [/D] [диск:][путь] CD [..] .. обозначает переход в родительский каталог.

Команда CD диск: отображает имя текущего каталога указанного диска. Команда CD без параметров отображает имена текущих диска и каталога. Параметр /D используется для одновременной смены текущих диска и каталога.

RD — удаление каталога. RMDIR [/S] [/Q] [диск:]путь RD [/S] [/Q] [диск:]путь /S Удаление дерева каталогов, т. е. не только

указанного катало- га, но и всех содержащихся в нем файлов и подкаталогов.

/Q Отключение запроса подтверждения при удалении дерева каталогов с помощью ключа /S.

COPY — копирование одного или нескольких файлов в другое место.

COPY [/D] [/V] [/N] [/Y | /-Y] [/Z] [/A | /B] источник [/A | /B] [+ источник [/A | /B] [+ ...]] [результат [/A | /B]] 6 источник Имена одного или нескольких копируемых файлов. /A Файл является текстовым файлом ASCII. /B Файл является двоичным файлом. /D Указывает на возможность создания зашифрованного файла результат Каталог и/или имя для конечных файлов. /V Проверка правильности копирования файлов. /N Использование, если возможно, коротких имен при копировании файлов, чьи имена не удовлетворяют стандарту 8.3. /Y Подавление запроса подтверждения на перезапись существующего конечного файла. /-Y Обязательный запрос подтверждения на перезапись существующего конечного файла. /Z Копирование сетевых файлов с возобновлением. Ключ /Y можно установить через переменную среды COPYCMD. Ключ /-Y командной строки переопределяет такую установку. По умолчанию требуется подтверждение, если только команда COPY не выполняется в пакетном файле. Чтобы объединить файлы, укажите один конечный и несколько исходных файлов, используя подстановочные знаки или формат «файл1+файл2+файл3+...». REN — переименование одного или нескольких файлов. RENAME [диск:][путь]имя_файла1 имя_файла2. REN [диск:][путь]имя_файла1 имя_файла2. Для конечного файла нельзя указать другой диск или каталог. DEL — удаление одного или нескольких файлов.

DEL [/P] [/F] [/S] [/Q] [/A[[:]атрибуты]] имена ERASE [/P] [/F] [/S] [/Q] [/A[[:]атрибуты]] имена

Для удаления сразу нескольких файлов используются подстановочные знаки. Если указан каталог, из него будут удалены все файлы. 17 /P Запрос на подтверждение перед удалением каждого файла. /F Принудительное удаление файлов, доступных только для чтения. /S Удаление указанных файлов из всех подкаталогов. /Q Отключение запроса на подтверждение при удалении фай- лов. /A Отбор файлов для удаления по атрибутам. атрибуты: - S Системные файлы - R Доступные только для чтения - H Скрытые файлы - A Файлы для архивирования - Префикс «-» имеет значение НЕ TYPE — вывод содержимого одного или нескольких текстовых файлов. TYPE [диск:][путь]имя_файла FOR — выполне-

ние указанной команды для каждого файла набора. FOR %переменная IN (набор) DO команда [параметры] %переменная – подставляемый параметр; (набор) – набор, состоящий из одного или нескольких файлов.

Допускается использование подстановочных знаков; команда – команда, которую следует выполнить для каждого файла; параметры – параметры и ключи для указанной команды. В пакетных файлах для команды FOR используется запись %%переменная вместо %переменная. Имена переменных учитывают регистр букв (%i отличается от %I). Добавление поддерживаемых вариантов команды FOR при включении расширенной обработки команд: FOR /D %переменная IN (набор) DO команда [параметры] 8 Если набор содержит подстановочные знаки, команда выполняется для всех подходящих имен каталогов, а не имен файлов. FOR /R [[диск:]путь] %переменная IN (набор) DO команда [параметры] Выполнение команды для каталога [диск:]путь, а также для всех подкаталогов этого пути. Если после ключа /R не указано имя катало- га, выполнение команды начинается с текущего каталога. Если вместо набора указана только точка (.), команда выводит список всех подкаталогов. FOR /L %переменная IN (начало, шаг ,конец) DO команда [пара- метры] Набор раскрывается в последовательность чисел с заданными началом, концом и шагом приращения. Так, набор (1,1,5) раскрывается в (1 2 3 4 5), а набор (5,-1,1) заменяется на (5 4 3 2 1)

FOR /F [«ключи»] %переменная IN (набор) DO команда [параметры] FOR /F [«options»] %variable IN («literal string») DO command [command-parameters] FOR /F [«options»] %variable IN ('command') DO command [command-parameters] или, если использован параметр usebackq: FOR /F [«options»] %variable IN (filenameset) DO command [command-parameters] FOR /F [«options»] %variable IN ('literal string') DO command [command-parameters] FOR /F [«options»] %variable IN (`command`) DO command [command-parameters]

Набор содержит имена одного или нескольких файлов, которые по очереди открываются, читаются и обрабатываются. Обработка состоит в чтении файла, разбивки его на отдельные строки текста и выделения из каждой строки заданного числа подстрок (в том числе нуля). Затем найденная подстрока используется в качестве значения переменной при выполнении основного тела цикла. По умолчанию ключ 9 /F выделяет из каждой строки файла первое слово, очищенное от окружающих его пробелов. Пустые строки в файле пропускаются. Не- обязательные парамет-

ры «ключи» служит для переопределения заданных по умолчанию правил обработки строк. Ключи представляют собой заключенную в кавычки строку, содержащую указанные параметры. Ключевые слова: eol=c — определение символа комментариев в конце строки (допускается задание только одного символа); skip=n — число пропускаемых при обработке строк в начале файла; delims=xxx — определение набора разделителей для замены за- данных по умолчанию пробела и знака табуляции; tokens=x,y,m-n — определение номеров подстрок, выделяемых из каждой строки файла и передаваемых для выполнения в тело цикла.

При использовании этого ключа создаются дополнительные переменные. Формат m-n представляет собой диапазон подстрок с номерами от m по n. Если последний символ в строке tokens= является звездочкой, создается дополнительная переменная, значением которой будет весь оставшийся текст в строке после обработки последней подстроки; usebackq — применение новой семантики, при которой строки, заключенные в обратные кавычки, выполняются как команды, строки, заключенные в прямые одиночные кавычки, являются строкой литера- лов команды, а строки, заключенные в двойные кавычки, используют- ся для выделения имен файлов в списках имен файлов.

Поясняющий пример: FOR /F "eol=; tokens=2,3* delims=," %%i in (myfile.txt) do @echo %%i %%j %%k — эта команда обрабатывает файл myfile.txt, пропускает все строки, которые начинаются с символа точки с запятой, и передает вторую и третью подстроки из каждой строки в тело цикла, причем подстроки разделяются запятыми и/или пробелами. В теле цикла переменная %i используется для второй подстроки, %j – для третьей, а %k получает все оставшиеся подстроки после третьей. Имена файлов, содержащие пробелы, необходимо заключать в двойные кавычки.

Для того чтобы использовать двойные кавычки, необходимо использовать параметр usebackq, иначе двойные кавычки будут восприняты как границы строки для обработки. 0 Переменная %i явно описана в инструкции for, а переменные %j и %k описываются неявно с помощью ключа tokens=. Ключ tokens= позволяет извлечь из одной строки файла до 26 подстрок, при этом, не допускается использование переменных больших чем буквы 'z' или 'Z'. Следует помнить, что имена переменных FOR являются глобальными, поэтому одновременно не может быть активно более 52 переменных.

Синтаксис команды FOR /F также позволяет обработать отдельную строку, с указанием параметра filenameset, заключенным в одиночные кавычки. Строка будет обработана как единая строка из входного файла. Наконец, команда FOR /F позволяет обработать строку вывода другой команды. Для этого следует ввести строку вызова команды в апострофах вместо набора имен файлов в скобках. Строка передается для выполнения обработчику команд CMD.EXE, а вывод этой команды записывается в память и обрабатывается так, как будто строка вывода взята из файла.

Например, следующая команда: FOR /F "usebackq delims==" %%i IN (`set`) DO @echo %%i — выведет перечень имен всех переменных среды, определенных в настоящее время в системе. SHIFT — изменение содержимого (сдвиг) подставляемых параметров для пакетного файла. SHIFT [/n] — команда SHIFT при включении расширенной об- работки команд поддерживает ключ /n, задающий начало сдвига пара- метров с номера n, где n может быть от 0 до 9.

Например, в следующей команде: SHIFT /2 %3 заменяется на %2, %4 на %3 и т.д., а %0 и %1 остаются без изменений. CALL — вызов одного пакетного файла из другого. CALL [диск:][путь]имя_файла [параметры] 1 параметры – набор параметров командной строки, необходимых пакетному файлу. CHOICE2 — ожидает ответа пользователя. CHOICE [/C[:]варианты] [/N] [/S] [/T[:]c,nn] [текст] /C[:]варианты — варианты ответа пользователя. По умолчанию строка включает два варианта: YN /N Ни сами варианты, ни знак вопроса в строке приглашения не отображаются. /S Учитывать регистр символов. /T[:]c,nn Ответ «c» выбирается автоматически после nn секунд ожидания текст Строка приглашения После выполнения команды переменная ERRORLEVEL приобретает значение, равное номеру выбранного варианта ответа. FC — сравнение двух файлов или двух наборов файлов и вывод различий между ними. FC [/A] [/C] [/L] [/LBn] [/N] [/OFF[LINE]] [/T] [/U] [/W] [/nnnn][диск1:][путь1]имя_файла1 [диск2:][путь2]имя_файла2 FC /B [диск1:][путь1]имя_файла1 [диск2:][путь2]имя_файла2

/A- Вывод только первой и последней строк для каждой группы различий. /B Сравнение двоичных файлов.

/C- Сравнение без учета регистра символов.

/L -Сравнение файлов в формате ASCII. /LBn Максимальное число несоответствий для заданного числа строк.

/N- Вывод номеров строк при сравнении текстовых файлов ASCII. /OFF[LINE] -Не пропускать файлы с установленным атрибу-

том «Автономный». /T Символы табуляции не заменяются эквивалентным числом пробелов.

/U-Сравнение файлов в формате UNICODE. 2 CHOICE — это внешняя команда. 22 /W Пропуск пробелов и символов табуляции при сравнении. /nnnn Число последовательных совпадающих строк, которое должно встретиться после группы несовпадающих. [диск1:][путь1]имя_файла1 Указывает первый файл или набор файлов для сравнения. [диск2:][путь2]имя_файла2 Указывает второй файл или набор файлов для сравнения. FIND — поиск текстовой строки в одном или нескольких файлах. FIND [/V] [/C] [/N] [/I] [/OFF[LINE]] «строка» [[диск:][путь]имя_файла[...]]

/V- Вывод всех строк, НЕ содержащих заданную строку. /C Вывод только общего числа строк, содержащих заданную строку. /N Вывод номеров отображаемых строк. /OFF[LINE] Не пропускать файлы с установленным атрибутом «Автономный».

/I -Поиск без учета регистра символов. «строка» Искомая строка. [диск:][путь]имя_файла. Один или несколько файлов, в которых выполняется поиск. Если путь не задан, поиск выполняется в тексте, введенном с клавиатуры либо переданном по конвейеру другой командой. SORT — осуществляет сортировку файла. SORT [/R] [/+n] [/M килобайтов] [/L язык] [/REC символов] [[диск1:][путь1]имя_файла1] [/T [диск2:][путь2]] [/O [диск3:][путь3]имя_файла3] /+n Задает число символов, n, до начала каждого сравнения. /+3 показывает, что каждое сравнение будет начинаться с третьего символа каждой строки. Строки меньше чем n символов собираются перед всеми остальными строками. По умолчанию, сравнение начинается с первого символа каждой строки. /L[OCALE] язык Перекрывает установленные в системе по умолчанию язык и раскладку заданными. Пока существует возможность только одного выбора: ««C»« – наиболее быстрый способ упорядочивания последовательности. Сортировка всегда идет без учета регистра. /M[EMORY] килобайтов Задает количество основной памяти, используемой для сортировки, в килобайтах. Размер памяти должен быть не менее 160КБ. /REC[ORD_MAXIMUM] символов Определяет максимальное число символов в записи (по умолчанию 4096, максимальное 65535). /R[EVERSE] Обратный порядок сортировки; т.е. сортировка идет от Я до А, и затем от 9 до 0. [диск1:][путь1]имя_файла1 Определяет имя сортируемого файла. Если оно опущено, то будет использоваться стандартный поток ввода. Явное задание сортируемого файла работает быстрее, чем перенаправление того же файла в качестве стандартного потока ввода. /T[EMPORARY]

[диск2:][путь2] Определяет путь к папке, содержащей рабочие файлы сортировки, в том случае, когда данные не помещаются в основной памяти. По умолчанию используется системная временная папка. /O[UTPUT] [диск3:][путь3]имя_файла3 определяет имя файла, в котором сохраняются отсортированные результаты. Если оно опущено данные записываются в стандартный поток вывода. Явное задание файла вывода работает быстрее, чем перенаправление стандартного потока вывода в этот же файл.

Задания для самостоятельного выполнения

При разработке учтите возможность неправильного запуска ваших программ (например, с недостаточным количеством аргументов) и предусмотрите вывод сообщения об ошибке и подсказки. При выполнении работы используйте данные методические указания и справочник THelp.

Вариант 1. Разработать командный файл создающий, копирующий или удаляющий файл, указанный в командной строке, в зависимости от выбранного ключа (замещаемого параметра) /n , /c , /d. 4

Вариант 2. Разработать командный файл создающий, копирующий или удаляющий каталог, указанный в командной строке, в зависимости от выбранного ключа (замещаемого параметра) /n , /c , /d.

Вариант 3. Разработать командный файл, добавляющий вводом с клавиатуры содержимое текстового файла (в начало или в конец в зависимости от ключей (замещаемого параметра) /b /e).

Вариант 4. Разработать командный файл, регистрирующий время своего запуска в файле протокола run.log и автоматически запускающий некоторую программу (например, антивирусную и т. п.) по пятницам или 13 числам.

Вариант 5. Разработать командный файл, копирующий произвольное число файлов заданных аргументами из текущего каталога в указываемый каталог.

Вариант 6. Разработать командный файл, который помещает список файлов текущего каталога в текстовый файл и в зависимости от ключа сортирует по какому-либо полю. Реализовать два варианта: с использованием только команды DIR, с использованием команд DIR и SORT.

Вариант 7. Разработать командный файл, который в интерактивном режиме мог бы дописывать в файл текст, удалять строки из файла, и распечатывать на экране содержимое файла.

Вариант 8. Разработать командный файл, который дописывал бы имя файла, полученного входным параметром в сам файл N количество раз. N – также задается параметром. Вариант 9. Разработать командный файл, который бы запускал бы какой-либо файл один раз в сутки. То есть, если файл запускается первый раз в сутки, то он запускает какой-либо файл. Если ваш файл уже запускали сегодня, то ваш файл ничего не делает. В работе используйте для сравнения дат команду FC.

Вариант 10. Разработать командный файл, который бы запускал бы какой-либо файл один раз в сутки. То есть, если файл запускается первый раз в сутки, то он запускает какой-либо файл. Если ваш файл 5 уже запускали сегодня, то ваш файл ничего не делает. Сравнение дат реализуйте через переменные, а не через файлы.

Вариант 11. Разработать командный файл, который получал в качестве параметра какое-либо имя, и проверял, определена ли токая переменная среды или нет, и выводил соответствующее сообщение.

Вариант 12. Разработать командный файл, который получал в качестве параметра какой-либо символ и в зависимости от второго параметра вырезал или сохранял в заданном файле все строки начинающиеся на этот символ.

Вариант 13. В некотором файле храниться список пользователей ПК и имя их домашних каталогов. Необходимо разработать программу, которая просматривает данный файл и в интерактивном режиме задает вопрос – копировать текущему пользователю (в его домашний каталог) какой-либо заданный файл (в качестве параметра) или нет. Если «Да» то программа копирует файл.

Вариант 14. Разработать командный файл, который бы выводил в зависимости от ключа на экран имя файла с самой последней или с самой ранней датой последнего использования.

Вариант 15. Разработать командный файл, который бы получал в качестве аргумента имя текстового файла и выводил на экран информацию о том, сколько символов, слов и строк в текстовом файле.

Вариант 16. Разработать командный файл (аналог команды tail в Unix). Командный файл печатает конец файла. По умолчанию – 10 последних строк. Явно можно задать номер строки, от которой печатать до конца.

Вариант 17. Разработать командный файл, который бы склеивал текстовые файлы, заданные в качестве аргументов, и сортировал бы строки результирующего файла в зависимости от

ключа по убыванию или по возрастанию.

Вариант 18. Разработать командный файл, который формировал бы ежемесячный отчет об изменениях в рабочем каталоге (файлы созданные, удаленные). 6

Вариант 19. Разработать командный файл, который формировал бы ежемесячный отчет об изменениях в рабочем каталоге (файлы измененные).

Вариант 20. Выполняющий в зависимости от ключа один из 3–х вариантов работы: - с ключом /n дописывает в начало указанных текстовых файлов строку с именем текущего файла; - с ключом /b создает резервные копии указанных файлов; - с ключом /d удаляет указанные файлы после предупреждения.