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?