

Лабораторная работа № 3

Использование консольных команд и пакетные файлы

Справочник по средствам командной строки Windows

Время от времени возникает необходимость автоматизировать обработку файлов. Обработка каждого файла отдельной командой, подаваемой вручную, не только отнимает время, но и приводит к трудно определяемым ошибкам, возникающим из-за естественной невнимательности оператора.

Во всех операционных системах семейства Microsoft Windows простейшим средством автоматизации обработки файлов (и каталогов) служат так называемые командные файлы. Использование командных файлов доступно всякому без особенной подготовки, вполне достаточно внимательности и здравого смысла.

Компиляторы и интерпретаторы

С помощью языка программирования создается не готовая программа, а только ее текст, описывающий ранее разработанный алгоритм. Чтобы получить работающую программу, надо этот текст либо автоматически перевести в машинный код (для этого служат *программы-компиляторы*) и затем использовать отдельно от исходного текста, либо сразу выполнять команды языка, указанные в тексте программы (этим занимаются *программы-интерпретаторы*).

Интерпретатор берет очередной оператор языка из текста программы, анализирует его структуру и затем сразу исполняет (обычно после анализа оператор транслируется в некоторое промежуточное представление или даже машинный код для более эффективного дальнейшего исполнения). Только после того как текущий оператор успешно выполнен, интерпретатор перейдет к следующему. При этом, если один и тот же оператор должен выполняться в программе многократно, интерпретатор всякий раз будет выполнять его так, как будто встретил впервые. Вследствие этого, программы, в которых требуется осуществить большой объем повторяющихся вычислений, могут работать медленно. Кроме того, для выполнения такой программы на другом компьютере там также должен быть установлен интерпретатор — ведь без него текст программы является просто набором символов.

По-другому, можно сказать, что интерпретатор моделирует некую виртуальную вычислительную машину, для которой базовыми инструкциями служат не элементарные команды процессора, а операторы языка программирования.

Компиляторы полностью обрабатывают весь текст программы (он иногда называется *исходный код*). Они просматривают его в поисках синтаксических ошибок (иногда несколько раз), выполняют определенный смысловой анализ и затем автоматически переводят (*транслируют*) на машинный язык — генерируют машинный код. Нередко при этом выполняется *оптимизация* с помощью набора методов, позволяющих повысить быстродействие программы (например, с помощью инструкций, ориентированных на конкретный процессор, путем исключения ненужных команд, промежуточных вычислений и т. д.). В результате законченная программа получается компактной и эффективной, работает в сотни раз быстрее программы, выполняемой с помощью интерпретатора, и может быть перенесена на другие компьютеры с процессором, поддерживающим соответствующий машинный код.

Основной недостаток компиляторов — трудоемкость трансляции языков программирования, ориентированных на обработку данных сложной структуры, часто заранее неизвестной или динамически меняющейся во время работы программы. Тогда в машинный код приходится вставлять множество дополнительных проверок, анализировать наличие ресурсов операционной системы, динамически их захватывать и освобождать, формировать и обрабатывать в памяти компьютера сложные объекты, что на уровне жестко заданных машинных инструкций осуществить довольно трудно, а для ряда задач практически невозможно.

С помощью интерпретатора, наоборот, допустимо в любой момент остановить работу программы, исследовать содержимое памяти, организовать диалог с пользователем, выполнить сколь угодно сложные преобразования данных и при этом постоянно контролировать состояние окружающей программно-аппаратной среды, благодаря чему достигается высокая надежность работы. Интерпретатор при выполнении каждого оператора проверяет множество характеристик операционной системы и при необходимости максимально подробно

информирует разработчика о возникающих проблемах. Кроме того, интерпретатор очень удобен для использования в качестве инструмента изучения программирования, так как позволяет понять принципы работы любого отдельного оператора языка.

В реальных системах программирования перемешаны технологии и компиляции, и интерпретации. В процессе отладки программа может выполняться по шагам, а результирующий код не обязательно будет машинным — он даже может быть исходным кодом, написанным на другом языке программирования (это существенно упрощает процесс трансляции, но требует компилятора для конечного языка), или промежуточным машиннонезависимым кодом абстрактного процессора, который в различных компьютерных архитектурах станет выполняться с помощью интерпретатора или компилироваться в соответствующий машинный код.

В состав многих операционных систем, в том числе, разработанных корпорацией Microsoft, входит командный процессор. Так называется программа, которая инициирует выполнение всевозможных действий в ответ на команды, вводимые пользователем с клавиатуры. Это Интерпретатор.

Для того чтобы запустить командный процессор:

1. Нажмите на кнопку **Пуск**. На экран будет выведено главное меню.
2. Выберите в главном меню пункт **Выполнить**. На экран будет выведено диалоговое окно **Запуск программы**.
3. В поле **Открыть** введите строку **cmd**.
4. Нажмите на кнопку **ОК**. На экран будет выведено окно командного процессора.

Текущий каталог. Абсолютные и относительные пути

При работе с файловыми командами исключительную важность приобретает понятие текущего каталога. Дело в том, что при указании файла в качестве параметра команды мы всегда используем один из двух возможных способов указания на них: либо абсолютный путь, либо относительный путь. В полном пути мы указываем все, начиная с диска (или сетевого имени компьютера), например **c:\vasya\mail\letter.txt**. Какой бы каталог ни оказался текущим в момент ввода команды, полный путь будет соответствовать одному и тому же файлу. Для относительного пути текущий каталог служит отправной точкой. Простейший случай относительного пути — имя файла. В контексте выполнения команды оно означает файл с таким именем, расположенный в текущем каталоге.

Для записи относительного пути к текущему каталогу существует условная запись **.** (точка). Для записи относительного пути к каталогу, в котором содержится текущий каталог, существует условная запись **..** (две точки).

Команда, показанная на следующем листинге, копирует все файлы из текущего каталога в каталог **reserve**, расположенный рядом с ним.

```
copy *.* .\reserve
```

Маски

Для задания группы папок и файлов используются маски

```
(*.doc)
(*.doc *.txt *.me)
(jan*.doc jan*.rpt feb*.doc feb*.rpt)
(ar??1991.* ap??1991.*)
```

Командная строка и команды

Для доступа к набору всех команд предназначена команда **help**. По этой команде на экран выводится список доступных команд. Для того чтобы получить описание конкретной команды, в качестве параметра команда **help** следует указать ее имя. Командная строка, показанная на следующем листинге, выводит на экран описание команды **for**.

```
help for
```

Если ввести команду **help**, то результат ее работы (т.н. выдача) не умещается на один экран. Та же проблема возникает с текстом описания команды **for**. Выдачу можно перенаправить в файл. Командная строка, показанная на следующем листинге, формирует файл **commands.txt**, содержащий список всех команд.

```
help > commands.txt
```

Для того чтобы сформировать файл с описанием команды **for**, надо дать такую команду (имя выходного файла можете сделать любым).

```
help for > for.txt
```

Help не единственный способ получить справку. Для любой команды из списка всех возможных в help всегда можно уточнить синтаксис по имени команды с ключом вида **/?**. Например, **For /?**. Выдаст справку о команде **for**.

copy — копирование одного или нескольких файлов;

del — удаление одного или нескольких файлов;

move — перемещение одного или нескольких файлов или каталогов;

rename (сокращенно **ren**) — переименование одного или нескольких файлов или каталогов;

xcopy — копирование дерева подкаталогов;

mkdir (сокращенно **md**) — создание каталога;

rmdir (сокращенно **rd**) — удаление каталога.

Одно из общих правил синтаксиса команд состоит в том, что при указании параметров сначала указывается источник, а потом результат. Например, если мы хотим переместить файл **text.txt** из каталога **mail** в каталог **drop**, мы должны ввести команду, приведенную на следующем листинге.

```
move mail\text.txt drop
```

Сначала что переместить, потом куда переместить.

Если мы хотим переименовать файл **igor.txt** в файл **misha.txt**, то команда должна быть записана так, как показано ниже.

```
ren igor.txt misha.txt
```

Сначала что переименовать, потом во что переименовать.

Командные файлы

При обработке большого количества файлов или при систематическом выполнении одних и тех же команд в командном процессоре предусмотрена возможность выполнения командных файлов. Командный файл — это текстовый файл, в котором набраны команды (или хотя бы одна команда).

В командном файле каждая команда занимает одну строку. Точнее, существует способ расположить одну команду на нескольких подряд идущих строках, для этого непосредственно перед каждым переводом строки следует поставить символ «крышка» **^**. (Необходимо, чтобы каждая «крышка» была последним символом в своей строке; после нее не должно быть пробелов и табуляций). Пример такой команды показан на следующем листинге.

```
if exist rt2.txt ^
copy rt2.txt ^
d:\doc\drafts
```

При выполнении командного файла командный процессор просматривает его сверху вниз от первой строки к последней и выполняет команды в том порядке, в котором их обнаруживает. Выполняет он в целом их так, как если бы мы каждую из них вводили вручную. В целом, потому что некоторые команды при вводе вручную и при выполнении из командного файла ведут себя немного по-разному.

Комментирование командного файла и его выдачи. Команды **echo** и **rem**

Командный файл, по существу, представляет собой программу, написанную на языке командного процессора операционной системы. Текст программы полагается снабжать комментариями, чтобы, вернувшись к нему некоторое время спустя, не вспоминать мучительно, для чего эта программа нужна, и как она устроена.

В системе команд для оформления комментариев предусмотрена команда **rem**. Это фиктивная команда, которая не предполагает выполнения каких бы то ни было действий, но позволяет написать в строке после своего имени произвольный текст. Причем командный процессор не воспринимает его как синтаксическую ошибку. Пример оформления командного файла комментариями показан на следующем листинге.

```
rem *****
rem Привет
rem *****

help copy
copy /?
```

При выполнении приведенного выше командного файла все команды будут выводиться на экран по мере их выполнения, что не всегда удобно. Выдачу команд можно отключить с помощью команды **@echo off**. Символ «собака» перед командой **echo** означает, то и сама эта команда должна выполняться в «молчаливом» режиме. С таким же успехом мы могли бы не пользоваться командой **echo off**, а поместить «собаку» перед каждой командой.

Передача командному файлу параметров

Предположим, мы хотим создать командный файл, который сначала формирует справку с описанием заданной пользователем команды, а потом загружает его для просмотра в текстовый редактор **edit**. Фокус в том, чтобы при очередном запуске командного файла каким-то образом сообщить ему, какая именно команда нас интересует на этот раз.

Для решения этой задачи предусмотрен механизм обработки параметров. Работает он довольно просто. Если при запуске командного файла пользователь указал несколько параметров, то в тексте командного файла первый из них мы обозначаем записью **%1**, второй записью **%2**, третий записью **%3** и т.д. Этими обозначениями мы пользуемся в тексте командного файла примерно так же, как в естественной речи местоимениями.

Текст командного файла, решающего поставленную задачу, приведен на следующем листинге. Обратите внимание на команду **help**. В качестве ее параметра ей передается первый параметр командного файла.

```
@echo off

rem Формируем файл с описанием команды,
rem имя которой передано параметром
help %1 > help.tmp

rem Загружаем файл описания в редактор Блокнот
edit help.tmp
```

Предположим, что мы присвоили этому командному файлу имя **show-help.bat**. Для того чтобы загрузить в текстовый редактор описание команды, например, **dir**, мы должны ввести команду следующим образом.

```
show-help.bat dir
```

Следующий командный файл создает каталог с именем, указанным в первом параметре, и записывает в него файл с текстом описания команды, указанной во втором параметре.

```
rem Пример командного файла с двумя параметрами
```

```
rem Создаем каталог с именем, заданным первым параметром
md %1
rem Создаем в нем файл с описанием команды,
rem заданной вторым параметром
help %2 > %1\%2.help
```

Что произойдет, если пользователь при запуске этого командного файла укажет не два, а четыре параметра? Ничего страшного, они ничему не помешают, просто не будут использованы. А что будет, если пользователь укажет только первый параметр? Вторым параметром окажется пустым. Эффект получится такой: командный файл будет выполнен, но так, как будто на месте записи **%2** ничего нет. Команда **help** сформирует список всех команд и поместит его в файл с пустым именем и расширением **.help**. Если же пользователь запустит этот файл, не указав ни одного параметра, то при попытке командного процессора выполнить команду **md** (напомним, она предназначена для создания каталога), мы получим сообщение о синтаксической ошибке, поскольку у команды **md** обязательно должен быть параметр.

Таким образом, использование параметров создает большие возможности, но может существенно усложнить дело. Для того чтобы командный файл всегда работал корректно, необходимо проверять корректность указания пользователем параметров и каким-то образом реагировать на неполные или неверные входные данные. Можно, конечно, этого и не делать, но некорректно работающий командный файл может наломать дров, особенно, если он предусматривает удаление или перезапись данных.

Средства перенаправления ввода-вывода

Текст с экрана можно записать в файл или, наоборот, из файла добавить текст можно на консоль средствами перенаправления.

Направите вход из файла (вместо клавиатуры):

```
< имя_файла
```

Направить выход в файл (вместо дисплея):

```
> имя_файла
```

При использовании символа **>** выходной файл перезаписывается. Для присоединения каждого последующего выходного файла к одноименному предыдущему используют **>>** вместо **>**:

```
>> имя_файла
```

Внутренние команды

ЕСНО

Управление «эхом» команд.

```
ЕСНО [ON | OFF | сообщение]
```

Отображает состояние «эха», включает или выключает «эхо» в пакетном файле или в командной строке или отображает сообщение.

Символ «собака» перед командой **echo** означает, то и сама эта команда должна выполняться в «молчаливом» режиме. С таким же успехом мы могли бы не пользоваться командой **echo off**, а поместить «собаку» перед каждой командой.

TITLE

Изменение заголовка окна командной строки.

```
TITLE [строка]
```

где строка — это будущий заголовок окна командной строки (записывается без кавычек).

После задания заголовка окна он может быть изменен только повторным вызовом команды **TITLE**. Использование данной команды может быть полезно в командных файлах. Так, выполнение примера, который приведен ниже, будет сопровождаться появлением соответствующей надписи в заголовке окна:

```
@ECHO OFF
TITLE Копируются файлы...
COPY \\Server\Share\*.doc C:\User\Common\*.doc
ECHO Копирование закончено.
TITLE Процесс завершен —
```

BREAK

Переключение режима контроля за нажатием клавиш Ctrl+C и Ctrl+Break.

```
BREAK [ON | OFF]
```

OFF — нажатие Ctrl+C или Ctrl+Break будет контролироваться только в процессе выполнения операций ввода-вывода с символьными устройствами — экраном, клавиатурой, последовательным портом и принтером.

ON — нажатие клавиш Ctrl+C или Ctrl+Break контролируется во время любой операции (в том числе дисковых операций ввода-вывода).

Команда BREAK (без параметров) отображает текущее состояние режима BREAK.

CD, CHDIR

Изменяет текущий каталог.

```
CD [диск:][путь]
CD [...]
CHDIR [диск:][путь]
CHDIR [...]
```

Команда CD (CHDIR) без параметров отображает имена текущих диска и каталога. Команда CD [диск:] (CHDIR [диск:]) отображает имя текущего каталога заданного диска. По команде CD [..](CHDIR [..]) совершается переход в родительский каталог.

CLS

Очищает дисплей и перемещает курсор в левый верхний угол экрана.

COPY

```
COPY [/D] [/V] [/N] [/Y | /-Y] [/Z] [/L] [/A | /B] источник
[/A | /B] [+ источник [/A | /B] [+ ...]] [результат [/A | /B]]
```

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 CRC в каждом секторе целевого файла при копировании 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.

/L Если источник является символической ссылкой, копирование ссылки вместо реального файла, на который указывает ссылка.

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+...". Копирует или соединяет один или более файлов.

DATE

Отображает или изменяет системную дату.

DATE [дата]

Если не введены никакие параметры, DATE отобразит текущие системную дату и время и попросит ввести новую дату. Введите новое значение даты или нажмите Enter, если не хотите менять дату.

DEL

Удаляет файл(ы) с диска.

DEL [диск:] [путь] имя_файла [/P]

Для удаления нескольких файлов можно использовать шаблоны имен файлов.

/P Запросить подтверждение (Y/N) удаления каждого файла.

DEL не удаляет скрытые файлы, системные файлы и файлы только для чтения.

DIR

Выводит список файлов и подкаталогов указанного (текущего) каталога.

DIR [диск:] [путь] [имя_файла]

DIR допускает использование шаблонов имени файла, по умолчанию *.*. Параметры могут быть установлены в переменной окружения DIRCMD. Текущие параметры команды дополняют опции, установленные в DIRCMD, или отменяют их, если задаются с префиксом —, например /—W.

EXIT

Покинуть вторичный командный процессор.

MD, MKDIR

Создает подкаталоги.

MD [диск:] путь MKDIR [диск:] путь

см. также RD.

MORE

Выводит информацию до заполнения экрана, затем делает паузу.

MORE [диск:] [путь] имя_файла

MORE < [диск:] [путь] имя_файла

команда | MORE [диск:] [путь] [имя_файла]

где команда — команда, вывод которой отображается на экране.

Нажатие любой клавиши приводит к выдаче на стандартное устройство вывода следующей порции информации. Это повторяется до тех пор, пока все входные данные не будут прочитаны.

PATH

Устанавливает или сообщает путь поиска исполняемых и пакетных файлов, не находящихся в текущем каталоге.

PATH [[диск:] путь1[; [диск:] путь2[; ...]]] PATH [;]

Команда PATH отобразит текущий путь поиска. Команда PATH ; очистит путь поиска.

PROMPT

Изменяет приглашение командной строки Windows.

PROMPT [текст]

где текст — текст нового приглашения командной строки.

Текст приглашения может содержать специальные строки в форме \$?, где ? может быть одним из следующих символов:

D Текущая дата в формате День, Месяц, Число. Год.

G >

L <

N Буква текущего диска.

P Текущие диск и путь.

Q =

T Текущее время в формате чч:мм:сс.

V Номер версии Windows.

CR/LF (переход к началу следующей строки).

\$ \$

PROMPT (без параметров) восстанавливает исходный вид приглашения.

RD, RMDIR

Удаляет один или несколько каталогов.

RD [диск:]путь RMDIR [диск:]путь

Прежде чем удалить подкаталог, в нем необходимо удалить все файлы и подкаталоги (и их файлы) (включая скрытые файлы и файлы только для чтения).

Можно использовать шаблоны * и ?. Нельзя удалять корневой каталог \, текущий каталог «.» или надкаталог «..».

SET

Переменной называется поименованное значение. В учебниках по программированию переменную обычно сравнивают с конвертом, на котором написано имя. Внутри конверта можно положить нечто, например, определенную сумму денег — это ее значение. Как и в случае с конвертом, значение переменной можно изменить.

Отображает, создает, редактирует или удаляет переменные окружения среды Windows.

SET [переменная = значение]

переменная — это имя переменной среды;

значение — значение переменной среды.

«SET переменная» отображает содержимое переменной.

«SET переменная =>» удаляет переменную из среды.

«SET переменная = значение» помещает или замещает значение.

Пример.

set photo_=c:\my_photo

После этого данную переменную можно использовать в своих командных файлах таким вот образом (используются также стандартные системные переменные):

dir %photo_%

Переменная	Тип	Описание
%ALLUSERSPROFILE%	Локальная	Возвращает размещение профиля «All Users».
%APPDATA%	Локальная	Возвращает используемое по умолчанию размещение данных приложений.
%CD%	Локальная	Возвращает путь к текущей папке.
%CMDCMDLINE%	Локальная	Возвращает строку команд, с помощью которой был запущен данный экземпляр Cmd.exe.
%CMDEXTVERSION%	Системная	Возвращает номер версии текущих расширений обработчика команд.
%COMPUTERNAME%	Системная	Возвращает имя компьютера.
%COMSPEC%	Системная	Возвращает путь к исполняемой командной оболочке.

Переменная	Тип	Описание
%DATE%	Системная	Возвращает текущие данные. Использует тот же формат, что и команда date /t .
%ERRORLEVEL%	Системная	Возвращает код ошибки последней использованной команды. Значение, не равное нулю, обычно указывает на наличие ошибки.
%HOMEDRIVE%	Системная	Возвращает имя диска локальной рабочей станции, связанного с основным каталогом пользователя. Задается на основании расположения основного каталога. Основным каталогом пользователя указывается в оснастке «Локальные пользователи и группы».
%HOMEPATH%	Системная	Возвращает полный путь к основному каталогу пользователя. Задается на основании расположения основного каталога. Основным каталогом пользователя указывается в оснастке «Локальные пользователи и группы».
%HOMESHARE%	Системная	Возвращает сетевой путь к общему основному каталогу пользователя. Задается на основании расположения основного каталога. Основным каталогом пользователя указывается в оснастке «Локальные пользователи и группы».
%LOGONSERVER%	Локальная	Возвращает имя контроллера домена, который проверял подлинность текущей сессии.
%NUMBER_OF_PROCESSORS%	Системная	Задаёт количество процессоров, установленных на компьютере.
%OS%	Системная	Возвращает имя операционной системы. При использовании Windows 2000 имя операционной системы отображается как Windows_NT.
%PATH%	Системная	Указывает путь поиска для исполняемых файлов.
%PATHEXT%	Системная	Возвращает список расширений файлов, которые рассматриваются операционной системой как исполняемые.
%PROCESSOR_ARCHITECTURE%	Системная	Возвращает архитектуру процессора. Значения: x86, IA64.
%PROCESSOR_IDENTIFIER%	Системная	Возвращает описание процессора.
%PROCESSOR_LEVEL%	Системная	Возвращает номер модели процессора, установленного на компьютере.
%PROCESSOR_REVISION%	Системная	Возвращает номер модификации процессора.
%PROMPT%	Локальная	Возвращает параметры командной строки для текущего интерпретатора. Создается командой Cmd.exe.

Переменная	Тип	Описание
%RANDOM%	Системная	Возвращает произвольное десятичное число от 0 до 32767. Создается командой Cmd.exe.
%SYSTEMDRIVE%	Системная	Возвращает имя диска, содержащего корневой каталог Windows XP (т. е. системный каталог).
%SYSTEMROOT%	Системная	Возвращает размещение системного каталога Windows XP.
%TEMP% и %TMP%	Системная и пользовательская	Возвращает временные папки, по умолчанию используемые приложениями, которые доступны пользователям, выполнившим вход в систему. Некоторые приложения требуют переменную TEMP, другие — переменную TMP.
%TIME%	Системная	Возвращает текущее время. Использует тот же формат, что и команда time /t . Создается командой Cmd.exe. Дополнительные сведения о команде time см. в разделе Time.
%USERDOMAIN%	Локальная	Возвращает имя домена, содержащего список учетных записей пользователей.
%USERNAME%	Локальная	Возвращает имя пользователя, выполнившего вход в систему.
%USERPROFILE%	Локальная	Возвращает размещение профиля для текущего пользователя.
%WINDIR%	Системная	Возвращает размещение каталога операционной системы.

TIME

Отображает или устанавливает системное время.

TIME [чч:мм:сс [A|P]]

чч час (0—23). мм минута (0—59). сс секунда (0—59). A AM (до полудня). P PM (после полудня).

TYPE

Отображает содержание указанного файла (файлов).

TYPE [диск:] [путь] имя_файла

VOL

Отображает метку (метки) и серийный номер тома диска.

VOL [диск:]

Если не указано имя диска, VOL отобразит метку и серийный номер тома текущего диска. Метки тома можно создавать, менять или удалять с помощью команды LABEL.

Команды пакетных файлов**Проверка условий и выбор вариантов. Команды if и goto**

Команда if позволяет выделять в командном файле группы команд, которые выполняются или не выполняются в зависимости от определенных условий. Для чего это нужно?

Проверка условия — почти необходимая мера при создании командных файлов, использующих параметры. Перед тем, как начинать работу, командный файл, вообще говоря, должен удостовериться в том, что ему передан корректный набор параметров. В противном случае велик риск, что он выполнится неверно или безрезультатно, а пользователю останется только

гадать, в чем же проблема. Более того, если командный файл удаляет, перемещает или перезаписывает какие-либо данные, то при некорректных параметрах он может даже нанести ущерб.

На следующем листинге показан командный файл компиляции хелп-файла. В начало командного файла добавлена проверка первого параметра на непустоту. Обратите внимание на такую особенность синтаксиса: для операции сравнения используется двоянный знак равенства. Если первый параметр оказывается непустым, срабатывает команда `goto`, которая «перепрыгивает» командный процессор к указанной метке. В данном случае имя этой метки `compile`. Обратите внимание, что там, где метка находится, ее имя предваряется двоеточием, а в команде `goto` нет. При пустом первом параметре командный процессор переходит к следующей строке, которая выдает сообщение об ошибке. А потом к следующей, которая перепрыгивает его в самый конец файла к метке с именем **finish**.

```
@echo off

rem Проверяем, задан ли параметр
if not "%1"==" " goto compile

rem Если параметр пуст, выдаем сообщение об ошибке
echo Не указано имя проекта хелп-файла
rem и переходим в конец командного файла
rem к метке finish
goto finish

rem Это метка с именем compile
:compile

rem Ниже расположены команды компиляции

rem Путь к компилятору хелп-файлов
set help_compiler="c:\Program Files\HTML Help Workshop\hhc.exe"

rem Путь к каталогу, в котором находятся проекты хелп-файлов
set project_path=e:\work\projects\help-projects

rem Вызываем компилятор для обработки конкретного проекта,
rem имя которого передаем в первом параметре
%help_compiler% %project_path%\%1.hpj

rem Это метка с именем finish
:finish
```

Скажем прямо, предложенный способ проверки параметра не самый удачный.

Во-первых, если пользователь по ошибке укажет в качестве параметра имя несуществующего файла, командный файл этим удовлетворится и предпримет попытку компиляции. Более правильный способ — проверить, существует ли такой файл в действительности. Для этого в языке команд MS-DOS предусмотрено специальное слово **exist**. Поэтому лучше было бы написать: `if exist %1.hpj goto compile`.

Во-вторых, активное использование команды **goto** (т.н. безусловного перехода) и меток сильно запутывают код. Технически они ничем не плохи, но отлаживать и сопровождать командный файл, написанный в таком стиле, довольно неудобно. Поэтому программисты издавна считают безусловный переход приемом нежелательным. Ниже показан более правильный, с точки зрения стиля программирования, структурированный вариант, в котором

используется конструкция **if...else**. Работает она так: если условие истинно, выполняются команды в скобках после **if**, а если ложно, то в скобках после **else**.

```
@echo off

rem Проверяем, задан ли параметр
if not exist %1.hpj (

    rem Если параметр пуст, выдаем сообщение об ошибке
    echo Такого проекта хелп-файла не существует.
) else (

    rem Ниже расположены команды компиляции

    rem Путь к компилятору хелп-файлов
    set help_compiler="c:\Program Files\HTML Help Workshop\hhc.exe"

    rem Путь к каталогу, в котором находятся проекты хелп-файлов
    set project_path=e:\work\projects\help-projects

    rem Вызываем компилятор для обработки конкретного проекта,
    rem имя которого передаем в первом параметре
    %help_compiler% %project_path%\%1.hpj

)
```

Обратите внимание на отступы от левого края. Они необязательны, но делают текст командного файла более читабельным.

Приведем еще один пример работы с проверками. Следующий командный файл создает каталог с именем **help-files** (предположим, для загрузки в него скомпилированных хелп-файлов). При этом, если каталог с таким именем уже существует (и в нем, вероятно, находятся старые хелп-файлы, которые не хотелось бы терять: вдруг новые окажутся хуже?), командный файл присваивает ему расширение bak. Но если каталог **help-files.bak** уже существовал, то командный файл его удаляет (будем считать, что одной резервной копии нам хватит).

```
if exist help-files.bak rd help-files.bak
if exist help-files ren help-files help-files.bak
md help-files
```

Массовая обработка файлов. Команда **for**

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

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

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

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

выполняется тело цикла, написанное после слова **do**. В приведенном примере это команда **echo**, которая выводит на экран текущее значение счетчика цикла.

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

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

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

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

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

Передача управления другому командному файлу. Команда **call**

Существует возможность вызвать из одного командного файла другой командный файл. Для этого служит команда **call**. Замечательно, переменные, заданные в вызывающем командном файле «видны» вызванному. И наоборот, после того, как вызванный файл закончит работу и вернет управление вызвавшему, последний будет «видеть» переменные, оставленные ему вызванным «в наследство». Это позволяет разработчику командных файлов действовать, например, следующим образом. Если несколько командных файлов должны пользоваться одними и теми же значениями, допустим, путями к каким-то файлам, их можно вынести в отдельный командный файл, который будет играть роль конфигурационного файла. Каждый рабочий командный файл будет начинаться вызовом конфигурационного. Выигрыш в том, что при изменении путей вносить изменения придется только в один конфигурационный файл, а не во множество рабочих.

«Конфигурационный» командный файл **config.bat**.

```
rem Путь к компилятору хелп-файлов
set help_compiler="c:\Program Files\HTML Help Workshop\hhc.exe"

rem Путь к каталогу, в котором находятся проекты хелп-файлов
set project_path=e:\work\projects\help-projects
«Рабочий» командный файл.
@echo off

rem Настраиваем переменные
call config.bat
```

```

rem Проверяем, задан ли параметр
if not exist %1.hpj (

    rem Если параметр пуст, выдаем сообщение об ошибке
    echo Такого проекта хелп-файла не существует.
) else (

    rem Ниже расположены команды компиляции

    rem Вызываем компилятор для обработки конкретного проекта,
    rem имя которого передаем в первом параметре
    %help_compiler% %project_path%\%1.hpj

)

```

PAUSE

Приостанавливает выполнение пакетного файла до нажатия клавиши.

```
pause
```

REM

Помещает комментарий в пакетный файл.

```
REM [комментарий]
```

Если ECHO находится в состоянии ON, комментарии появятся на экране.

Внешние команды**ATTRIB**

Изменяет или отображает атрибуты файла или каталога.

```
ATTRIB [+R I -R] [+A | -A] [+S | -S] [+H ] -H]
[[диск:][путь]имя_фай-ла] [/S]
```

+A Установить атрибут «архивный».

–A Отменить атрибут «архивный». +H Установить атрибут «скрытый».

–H Отменить атрибут «скрытый».

+R Установить атрибут «только для чтения».

–R Отменить атрибут «только для чтения».

+S Установить атрибут «системный».

–S Отменить атрибут «системный».

/S Обработать файлы во всех подкаталогах указанного пути.

CHKDSK

Проверяет и исправляет дисковые каталоги, файлы и таблицу размещения файлов, отображает состояние диска и оперативной памяти.

```
CHKDSK [диск:][[путь]имя_файла] [/F] [/V]
```

Если задано имя файла (допустимо использование шаблонов), CHKDSK будет проверять дисковое пространство, занимаемое этим файлом (файлами), на степень фрагментации и отображать отчет о состоянии файла (файлов).

/F Исправить ошибки в каталоге или в таблице размещения файлов.

/V Отображать полные имена всех файлов на указанном диске.

Команда CHKDSK без параметров проверяет текущий диск.

DELTREE

Удаляет указанный каталог и все его подкаталоги с содержащимися в них файлами. Удаляет один или несколько файлов и каталогов.

```
DELTREE [/Y] [диск:]путь [[диск:]путь[...]]
```

В одной командной строке можно задать несколько каталогов.

[Диск:]путь Имя удаляемого каталога.

/Y Подавляет запрос на подтверждение удаления каталогов [Y/N].

DISKCOPY

Выполняет точное копирование дискет.

DISKCOPY [диск1; [диск2:]] [/1] [/V] [/M]

Эта команда позволяет копировать гибкие диски только одного типа. Параметры «диск!:)» и «диск2:)» могут указывать на один и тот же дисковод.

Не работает с дисководами, «порожденными» командой SUBST.

/I Копировать только первую сторону дискеты.

/V Гарантирует чтение каждого сектора после записи данных на него.

/M Использовать для временного хранения данных только базовую оперативную память, при этом дискета высокой плотности копируется в несколько проходов.

EDIT

EDIT [/B] [/S] [/H] [/R] [/<nnn>] [[[диск:] [путь] имя_файла] [...]]

EDIT — многооконный редактор для использования с текстовыми и двоичными файлами.

Может открывать до 9 файлов одновременно, загружать файлы командой с использованием шаблонов имен (типа EDIT *.*).

/B Работа в монохромном режиме на цветном дисплее.

/S Использование только коротких имен файлов.

/H Запускает редактор EDIT с максимальным числом строк, допустимым для этого экрана.

/R Загрузка файлов в режиме «только для чтения».

/<nnn> Загрузка двоичных файлов с разбивкой на строки по ppp символов.

FC

Сравнивает два файла или два ряда файлов для выявления различий.

FC [/A][/C][/L][/LB n][/W][/N][/T][/nn] [диск:] [путь] первый
[диск:] [путь] второй

FC выполняет построчное или побайтное сравнение файлов «первый» и «второй». По умолчанию FC выполняет построчное сравнение для всех типов файлов, кроме EXE, COM, SYS, OBJ, LIB или BIN.

/A Сократить отображение текстовых файлов.

/B Побайтное сравнение.

/C Игнорировать регистр для алфавитных символов.

/L Построчное сравнение файлов.

/LB n Прервать выполнение, если файлы имеют более чем n последовательно различных строк. По умолчанию n=100.

/N Нумеровать строки.

/T Не заменять символы табуляции (иначе символ табуляции заменяется 8 пробелами).

/W Сжимать пробелы, символы табуляции и пустые строки до одиночного пробела.

/пп Минимальное число строк, которые должны совпадать, для того чтобы FC принял их как одинаковые. По умолчанию равно 3.

Все параметры должны предшествовать именам файлов. Параметр /B нельзя использовать с любыми другими параметрами.

FIND

Ищет в файлах заданную последовательность символов.

FIND [/V][/C][/N][/I] «строка» [диск:] [путь] имя_файла.

/V Выдавать на экран все строки, которые не содержат указанную последовательность символов.

/C Выдавать порядковый номер строки, содержащей искомую последовательность символов.

/N Выдавать номер каждой строки, в которой есть искомая последовательность символов.

/I Игнорировать различия регистров.

LABEL

Создает, изменяет или удаляет метку тома диска.

```
LABEL [диск:] [метка]
```

Метка тома может содержать до 11 символов.

MOVE

Перемещает файлы в другой каталог или на другой диск, переименовывает каталоги.

```
MOVE [/Y|/Y][диск:] [путь] имя_файла [диск:] [путь] [имя_файла]
```

Команда MOVE перемещает указанный файл (файлы) по адресу файла, указанного последним. Если адресат уже существует, он удаляется. Нельзя перемещать файл сам в себя. Если задано более одного файла-источника, адресат должен быть каталогом, и файлы перемещаются в каталог под собственными именами. Если адресат не является каталогом, MOVE сообщит об ошибке и закончит работу.

Команда MOVE, вызванная из командной строки, предупреждает об опасности перезаписывания одноименного файла-адресата и не делает этого при вызове из пакетного файла.

/Y Никогда не выдавать запрос на подтверждение перезаписи одноименного файла.

/Y Всегда запрашивать подтверждение на перезапись одноименного файла-адресата.

XCOPY

Копирует файлы и каталоги, включая подкаталоги.

```
xcopy источник [результат] [/w] [/p] [/c] [/v] [/q] [/f] [/l] [/g]
[/d[:мм-дд-гггг]] [/u] [/i] [/s [/e]] [/t] [/k] [/r] [/h]
[{/a|/m}] [/n] [/o] [/x] [/exclude:файл1[+[файл2]][+[файл3]]]
[/Y|/Y] [/z]
```

Параметры

источник — обязательный параметр. Задает местонахождение и имена файлов для копирования. Параметр должен задавать или диск, или путь.

результат — задает место, куда будут скопированы файлы. Параметр может включать имя диска с двоеточием, имя каталога, имя файла или их комбинацию.

/w — выводит следующее сообщение с ожиданием подтверждения начала копирования: "Нажмите любую клавишу, чтобы начать копирование файлов"

/p — запрашивает подтверждение при создании каждого файла-результата.

/c — игнорирует ошибки.

/v — проверяет каждый скопированный файл на соответствие его оригиналу. Данная команда не используется в Windows XP. Она предназначена для обеспечения совместимости с файлами MS-DOS.

/q — отменяет вывод на экран сообщений команды xcopy.

/f — выводит имена исходных файлов и файлов-результатов в процессе копирования.

/l — отображает список копируемых файлов.

/g — создает незашифрованные файлы-результаты.

/d[:мм-дд-гггг] — копирует только файлы, измененные не ранее заданной даты. Если не включить значение мм-дд-гггг, команда xcopy копирует все файлы-источники, которые новее существующих файлов-результатов. Эта возможность позволяет обновлять только измененные файлы.

/u — копирует (обновляет) только те файлы-источники, которые уже существуют в каталоге результата.

/i — если источником является каталог или источник содержит подстановочные знаки и результат не существует, команда xcopy считает, что результат — это имя каталога, и создает новый каталог. Затем xcopy копирует все указанные файлы в новый каталог. По умолчанию команда xcopy запрашивает подтверждение, является ли параметр результат каталогом или файлом.

/s — копирует каталоги и подкаталоги, если они не пусты. Если параметр /s не задан, команда xcopy будет работать только с одним каталогом.

/e — копирует все подкаталоги, включая пустые. Параметр /e используется с параметрами /s и /t.

/t — копирует только структуру подкаталога (т.е. дерево), а не файлы. Для копирования пустых каталогов следует задать ключ /e.

/k — копирует файлы с атрибутом "только для чтения" с сохранением этого атрибута для скопированных файлов, оригиналы которых имеют этот атрибут. По умолчанию команда хсору удаляет атрибут "только для чтения".

/r — копирует файлы с атрибутом "только для чтения".

/h — копирует системные и скрытые файлы. По умолчанию команда хсору не копирует системные и скрытые файлы.

/a — копирует только те файлы, которые имеют установленный атрибут "архивный". При использовании параметра /a атрибут "архивный" у исходных файлов не меняется. Сведения об установке атрибутов с помощью команды attrib см. по ссылке "См. также".

/m — копирует только те файлы, которые имеют установленный атрибут "архивный". В отличие от параметра /a, параметр /m очищает атрибут "архивный" у скопированных файлов. Сведения об установке атрибутов с помощью команды attrib см. по ссылке "См. также".

/n — копирует с использованием коротких имен файловой системы NTFS. Параметр /n требуется при копировании из файловой системы NTFS в файловую систему FAT или когда на диске-результате требуется использование соглашения об именах файлов как в файловой системе FAT (8.3). Файлы могут записываться в файловую систему FAT или NTFS.

/o — копирует сведения о принадлежности файлов и об избирательной таблице управления доступом (DACL).

/x — копирует сведения о параметрах аудита файла и о системной таблице управления доступом (SACL) (подразумевается наличие /r).

/exclude:файл1+[файл2][+файл3] — определяет список файлов, содержащих строки.

При соответствии выведенной строки части пути копируемого файла он исключается из процесса копирования. Например, если указана строка "\Obj\", исключаются все файлы, расположенные в каталоге "Obj". Например, если указана строка ".obj", исключаются все файлы с расширением .obj.

/y — устраняет выдачу запроса на подтверждение перезаписи существующего конечного файла.

/-y — выдает запрос на подтверждение перезаписи существующего конечного файла. Можно применять параметр /y в переменной среде COPYCMD. Эта настройка может быть переопределена использованием параметра /-y в командной строке. По умолчанию если команда сору выполняется не в пакетной программе, при замене требуется подтверждение.

/z — копирует по сети в режиме перезапуска. Если во время фазы копирования теряется сетевое подключение (например, если сервер переходит в автономный режим, разрывая подключение), копирование возобновляется после восстановления подключения. Использование параметра /z команды вызывает также отображение доли (в процентах) завершенной операции копирования для каждого файла.

При копировании при помощи команды хсору файлов на том, не поддерживающий шифрованную файловую систему (EFS), возникнет ошибка. Следует предварительно расшифровать файлы или копировать их на том, поддерживающий EFS.

Эту команду можно использовать и для объединения нескольких файлов, для этого укажите один файл-результат, но несколько файлов-источников (с помощью подстановочных знаков или формата файл1+файл2+файл3).

Если при копировании не указывать параметр результат, хсору будет копировать файлы в текущий каталог. В том случае, когда параметр результат не содержит существующий каталог или не заканчивается обратной чертой (\), выводится следующее сообщение:

Что означает destination:

имя файла или каталога

(F = файл, D = каталог)?

Нажмите F, если файл или файлы должны копироваться в файл. Нажмите D, если файл или файлы должны копироваться в каталог.

Для устранения вывода этого сообщения используйте параметр /i, в результате чего команда хсору предполагает, что результат является каталогом, если источник представляет собой несколько файлов или каталогов.

Следует отметить, что команда хсору создает файлы с установленным атрибутом "архивный" независимо от состояния этого атрибута у исходных файлов.

Для анализа кодов завершения, выведенных командой хсору, можно использовать параметр уровень_ошибки в командной строке if пакетных программ. В следующей таблице перечислены коды завершения с кратким описанием.

Код завершения — Описание

- 0 — Файлы скопированы без ошибок
- 1 — Файлы для копирования не найдены
- 2 — Нажата комбинация CTRL+C для остановки команды хсору
- 4 — Возникла ошибка инициализации. Недостаточно места в памяти или на диске, введено неверное имя диска или неверный синтаксис вызова команды
- 5 — Диск защищен от записи

Лабораторное задание

Выполнить следующее задание, базируясь на основе только консольных команд.

1. Создать на диске **D:** папку с именем «**lab2**».
2. Добавить системную переменную с именем **lab2** для этой папки.
3. Сохранить файл справки по консольным командам в файл **commands.txt**.
4. Переместить файл **commands.txt** в папку, описанную переменной **%lab2%**.
5. Открыть не редактирование файл **commands.txt** командой **edit**.
6. Перейти в файле на строку, соответствующую номеру варианта. Отсчитать от данной строки 10 команд.
7. Для выбранного набора команд создать текстовые файлы справки с именем, совпадающим с названием команды, расширением **.hlp** и атрибутом «архивный».
8. В папке, описанной переменной **%lab2%** создать вложенную папку с именем **help**.
9. Создать текстовый документ в папке **help** с таким же именем и расширением **.txt**.
10. Поместить содержимое всех файлов справки (*.hlp) в документ **help.txt**.
11. Подготовить пакетный файл для успешного выполнения указанных действий.
12. Осуществить проверку первого параметра пакетного файла на не пустоту с помощью условного оператора **IF**, ключевого слова **exist** и метки перехода. В случае если параметр пуст, сформировать сообщение об ошибке.
13. Выполнить пункт 10 необходимо с помощью цикла **FOR**.

Перечень команд для быстрого запуска системных компонентов ОС Windows

Таблица 1

Русское название элемента	Английское название элемента	Команда
Ftp-протокол (командная строка)	Ftp-protocol (command prompt)	ftp
Iexpress Wizard (не переведена)	Iexpress Wizard	iexpress
Internet Explorer	Internet Explorer	ieexplore
Paint	Paint	mspaint
Windows Firewall	Брандмауэр Windows	firewall.cpl
Wordpad	Wordpad	write
Администратор источников данных ODBC	ODBC Data Source Administrator	odbcad32
Администрирование	Administrative Tools	control admintools
Архивация и восстановление	Backup and Restore	sdclt
Блокнот	Notepad	notepad
Восстановление системы	System Restore	rstrui
Выйти из системы	Log Out Of Windows	logoff
Дата и Время	Date and Time	timedate.cpl
Дефрагментация диска	Disk Defragmenter	dfrgui
Диспетчер авторизации	Authorization Manager	azman.msc
Диспетчер задач Windows	Windows Task Manager	taskmgr
Диспетчер проверки драйверов	Driver Verifier Manager	verifier
Диспетчер устройств	Device Manager	devmgmt.msc
Диспетчер устройств	Device Manager	hdwwiz.cpl
Дополнительные часы	Additional Clocks	control timedate.cpl,,1
Завершение работы Windows	Shuts Down Windows	shutdown /s
Записки	Sticky Note	StikyNot
Защита БД учетных записей Windows	Securing the Windows Account Database	syskey
Звук	Sound	mmsys.cpl
Звуки (звуковая схема)	Sounds (sound theme)	control mmsys.cpl,,2
Звукозапись	Sound Recorder	soundrecorder
Игровые устройства	Game Controllers	joy.cpl
Инициализация оборудования безопасности для TPM	Initialize the TPM security hardware	TpmInit
Калибровка цветов экрана	Display Color Calibration	dccw
Калькулятор	Calculator	calc
Командная строка	Command Prompt	cmd
Компоненты Windows	Windows Features	OptionalFeatures
Консоль управления (MMC)	Microsoft Management Console	mmc
Конфигурация системы	System Configuration	msconfig
Локальная политика безопасности	Local Security Policy	secpol.msc
Локальные пользователи и группы	Local Users and Groups	lusrmgr.msc
Мастер загрузки изображений Windows	Windows Picture Acquisition Wizard	wiaacmgr
Мастер создания общих ресурсов	Create A Shared Folder Wizard	shrpubw
Мастер установки драйверов устройств	Driver Package Installer	dpinst
Мастер установки оборудования	Add Hardware Wizard	hdwwiz
Микшер громкости	Sound Volume	sndvol
Монитор ресурсов	Resource Monitor	resmon
Настройка доступа программ и умолчаний	Set Program Access and Computer Defaults	control appwiz.cpl,,3
Настройка Контроля Учетных Записей	User Account Control Settings	UserAccountControlSettings
Ножницы	Snipping Tool	snippingtool
Общие папки	Shared Folders	fsmgmt.msc
Очистка диска	Disk Cleanup Utility	cleanmgr
Панель управления	Control Panel	control
Папка "Fonts"	Fonts Folder	fonts
Папка "Загрузки"	"Downloads" Folder	Downloads
Параметры папок	Folder Options	control folders
Перезагрузка	Restart Windows	shutdown /r
Перенос принтеров	Printer Migration	PrintBrmUi

Русское название элемента	Английское название элемента	Команда
Перо и сенсорные устройства	Pen and Touch	TabletPC.cpl
Персонализация	Personalization	control desktop
Планировщик заданий	Task Scheduler	control schedtasks
Подключение к удаленному рабочему столу	Remote Desktop Connection	mstsc
Получение программ	Get Programs	control appwiz.cpl,,1
Проверка диска	Check Disk Utility	chkdsk
Проверка и восстановление системных файлов	System File Checker (Scan and Repair)	sfc /scannow
Проверка подписи файла	File Signature Verification	sigverif
Проводник	Windows Explorer	explorer
Программа DiskPart	Disk Partition Manager	diskpart
Программы и компоненты	Programms and Features	appwiz.cpl
Просмотр событий	Event Viewer	eventvwr.msc
Разрешение экрана	Screen Resolution	desk.cpl
Редактор личных знаков	Private Character Editor	eudcedit
Редактор локальной групповой политики	Local Group Policy Editor	gpedit.msc
Редактор реестра	Registry Editor	regedit
Редактор реестра	Registry Editor	regedt32
Редактор титульных страниц факсов	Fax Cover Sheet Editor	fxscover
Результирующая политика	Resultant Set of Policy	rsop.msc
Сведения о системе	System Information	msinfo32
Свойства системы	System Properties	sysdm.cpl
Свойства системы: Дополнительно	System Properties: Advanced	SystemPropertiesAdvanced
Свойства системы: Защита системы	System Properties: System Protection	SystemPropertiesProtection
Свойства системы: Оборудование	System Properties: Hardware	SystemPropertiesHardware
Свойства системы: Удаленный доступ	System Properties: Remote	SystemPropertiesRemote
Свойства: Инициатор iSCSI	iSCSI Initiator Properties	iscsicpl
Свойства: Интернет	Internet Properties	inetcpl.cpl
Свойства: Клавиатура	Keyboard Properties	control keyboard
Свойства: Мышь	Mouse Properties	control mouse
Свойства: Мышь	Mouse Properties	main.cpl
Свойства: Мышь: Параметры указателя	Mouse Properties: Pointer Options	control main.cpl,,2
Свойства: Мышь: Указатели (схема)	Mouse Properties: Pointers	control main.cpl,,1
Сертификаты	Certificates	certmgr.msc
Сетевые подключения	Network Connections	control netconnections
Сетевые подключения	Network Connections	ncpa.cpl
Системный монитор	Performance Monitor	perfmon
Служба индексирования	Indexing Service	ciadv.msc
Службы компонентов	Component Services	dcomcnfg
Службы компонентов	Component Services	comexp.msc
Совместимость программы	Program Compatibility	msdt.exe -id PCWDiagnostic
Создать диск восстановления системы	Create a system repair disk	recdisc
Соседние пользователи	People Near Me	collab.cpl
Сохранение имен пользователей и паролей	Stored User Names and Passwords	credwiz
Средство диагностики DirectX	Direct X Troubleshooter	dxdiag
Средство диагностики службы технической поддержки	Microsoft Support Diagnostic Tool	msdt
Средство калибровки дигитайзера	Digitalizer Calibration Tool	tabcal
Средство настройки текста ClearType	ClearType Text Tuner	cttune
Средство просмотра XPS	XPS Viewer	xpsrchvw
Средство записи действий по воспроизведению неполадок	Problem Steps Recorder	psr
Таблица символов	Character Map	charmap
Телефон и модем	Phone and Modem	telephon.cpl
Удаленный помощник Windows	Windows Remote Assistance	msra
Управление дисками	Disk Management	diskmgmt.msc
Управление компьютером	Computer Management	compmgmt.msc
Управление печатью	Print Management	printmanagement.msc
Управление цветом	Color Management	colorcpl
Установка или удаление языков интерфейса	Install or uninstall display languages	lpksetup
Устройства и принтеры	Devices and Printers	control printers

Русское название элемента	Английское название элемента	Команда
Учетные записи пользователей	User Accounts	Netplwiz
Цвет и внешний вид окна	Window Color and Appearance	control color
Центр мобильности Windows	Windows Mobility Center	mbctr
Центр поддержки	Action Center	wscui.cpl
Центр синхронизации	Sync Center	mobsync
Центр специальных возможностей	Ease of Access Center	utilman
Шифрующая файловая система (EFS)	Encryption File System	rekeywiz
Шрифты (добавление или удаление)	Fonts	control fonts
Экран (размер текста)	Display (size of text)	dpiscaling
Экранная клавиатура	On-Screen Keyboard	osk
Экранная лупа	Magnifier	magnify
Экранный диктор	Microsoft Narrator	narrator
Электропитание	Power Options	powercfg.cpl
Электропитание: Дополнительные параметры	Power Options: Advanced Settings	control powercfg.cpl,,1
Элемент управления WMI	Windows Management Infrastructure	wmimgmt.msc
Язык и региональные стандарты	Region and Language	intl.cpl
Язык и региональные стандарты: Дополнительно	Region and Language: Administrative	control intl.cpl,,3
Язык и региональные стандарты: Языки и клавиатуры	Region and Language: Keyboards and Languages	control intl.cpl,,2