Лабораторная работа №1

Командный интерфейс ОС Windows.

Теоретические сведения

1. Оболочка командной строки Windows. Интерпретатор Cmd.exe

В операционной системе Windows, как и в других операционных системах, интерактивные (набираемые с клавиатуры и сразу же выполняемые) команды выполняются с помощью так называемого командного интерпретатора, иначе называемого командным процессором или оболочкой командной строки (command shell). Командный интерпретатор или оболочка командной строки — это программа, которая, находясь в оперативной памяти, считывает набираемые вами команды и обрабатывает их.
Для запуска командного интерпретатора (открытия нового сеанса командной строки) можно выбрать пункт Выполнить… (Run) в меню Пуск (Start), ввести имя файла Cmd.exe и нажать кнопку OK

1.1 Синтаксис командной строки, перенаправление ввода – вывода

Файловая система имеет древовидную структуру и имена файлов задаются в формате *[диск:] [путь\]имя_файла*. Если путь начинается с символа «\», то маршрут вычисляется от корневого каталога – иначе от текущего.
Например, c:123.txt задает файл 123.txt в текущем каталоге, c:\123.txt – в корневом, а DOC\123.txt – в подкаталоге DOC текущего каталога.
Существуют особые обозначения для текущего каталога (точка «.») и трех его верхних уровней (две точки «..» - родительский, три «…» - второго уровня и, наконец, четыре «….» - третьего уровня).
Например, для текущего каталога C:\Windows\Media\Office97 путь к файлу autoexec.bat в корневом каталоге диска C: может быть записан в виде ….\autoexec.bat.
В именах файлов (но не дисков или каталогов) можно применять так называемые групповые символы или шаблоны: ? (вопросительный знак) и * (звездочка). Символ * в имени файла означает произвольное количество любых допустимых символов, символ ? — один произвольный символ или его отсутствие. Скажем, под шаблон `text??1.txt` подходят, например, имена text121.txt и text11.txt, под шаблон `text*.txt` — имена text.txt, textab12.txt, а под шаблон `t e x t . *` — все файлы с именем text и произвольным расширением.
Например, `DIR /? > helpdir.txt` выведет справку по команде DIR в файл. Символ «>>» позволяет не создавать файл заново, а дописать в него. По аналогии символ «<» позволяет читать данные не с клавиатуры, а с файла. Например, `DATE < date.txt` ввод новой даты из файла.

1.2 Переменные окружения

При загрузке ОС Windows в оперативной памяти постоянно хранится набор т.н. *переменных окружения* (environment variables). Хотя в Windows есть более совершенный способ для хранения системных значений – реестр, многие программы по-прежнему используют переменные окружения. Наиболее важные переменные хранят системный путь для поиска (PATH), каталог запуска Windows (WINDIR), место хранения временных файлов (TEMP) и многое другое.
Переменные устанавливаются с помощью команды
`SET [переменная=[строка]]`

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

1.3 Внутренние и внешние команды. Структура команд

Некоторые команды распознаются и выполняются непосредственно самим командным интерпретатором — такие команды называются внутренними (например, `COPY` или `DIR`) Другие команды операционной системы представляют собой отдельные программы, расположенные по умолчанию в том же каталоге, что и Cmd.exe, которые Windows загружает и выполняет аналогично другим программам. Такие команды называются внешними (например, `MORE` или `XCOPY`).

Рассмотрим структуру самой командной строки и принцип работы с ней. Для того, чтобы выполнить команду, вы после приглашения командной строки (например, `C:\>`) вводите имя этой команды (регистр не важен), ее параметры и ключи (если они необходимы) и нажимаете клавишу <Enter>. Например:

```
C:\>COPY C:\myfile.txt A:\ /V
```

Имя команды здесь — `COPY`, параметры — `C:\myfile.txt` и `A:\`, а ключом является `/V`. Отметим, что в некоторых командах ключи могут начинаться не с символа /, а с символа – (минус), например, `-V`.

Многие команды Windows имеют большое количество дополнительных параметров и ключей, запомнить которые зачастую бывает трудно. Большинство команд снабжено встроенной справкой, в которой кратко описываются назначение и синтаксис данной команды. Получить доступ к такой справке можно путем ввода команды с ключом `/?`.

1.4 Условное выполнение и группировка команд

В командной строке Windows NT/2000/XP можно использовать специальные символы, которые позволяют вводить несколько команд одновременно и управлять работой команд в зависимости от результатов их выполнения. С помощью таких символов условной обработки можно содержание небольшого пакетного файла записать в одной строке и выполнить полученную составную команду.

Используя символ амперсанта `&`, можно разделить несколько утилит в одной командной строке, при этом они будут выполняться друг за другом. Например, если набрать команду `DIR & PAUSE & COPY /?` и нажать клавишу <Enter>, то вначале на экран будет выведено содержимое текущего каталога, а после нажатия любой клавиши — встроенная справка команды `COPY`.

Символ `^` позволяет использовать командные символы как текст, то есть при этом происходит игнорирование значения специальных символов. Например, если ввести в командной строке

```
ECHO Абв & COPY /?
```

и нажать клавишу <Enter>, то произойдет выполнение подряд двух команд: `ECHO Абв` и `COPY /?` (команда `ECHO` выводит на экран символы, указанные в командной строке после нее). Если же выполнить команду

```
ECHO Абв ^& COPY /?
```

то на экран будет выведено

```
Абв & COPY /?
```

В этом случае просто выполняется одна команда `ECHO` с соответствующими параметрами.

Условная обработка команд в Windows осуществляется с помощью символов `&&` и `||` следующим образом. Двойной амперсант `&&` запускает команду, стоящую за ним в командной строке, только в том случае, если команда, стоящая перед амперсантами была выполнена успешно. Например, если в корневом каталоге диска C: есть файл plan.txt, то

выполнение строки `TYPE C:\plan.txt && DIR` приведет к выводу на экран этого файла и содержимого текущего каталога. Если же файл C:\plan.txt не существует, то команда `DIR` выполняться не будет.

Два символа `||` осуществляют в командной строке обратное действие, т.е. запускают команду, стоящую за этими символами, только в том случае, если команда, идущая перед ними, не была успешно выполнена. Таким образом, если в предыдущем примере файл C:\plan.txt будет отсутствовать, то в результате выполнения строки `TYPE C:\plan.txt || DIR` на экран выведется содержимое текущего каталога.

Отметим, что условная обработка действует только на ближайшую команду, то есть в строке

TYPE C:\plan.txt && DIR & COPY /?

команда COPY /? запустится в любом случае, независимо от результата выполнения команды TYPE C:\plan.txt.

Несколько утилит можно сгруппировать в командной строке с помощью скобок. Рассмотрим, например, две строки:

TYPE C:\plan.txt && DIR & COPY /?

TYPE C:\plan.txt && (DIR & COPY /?)

В первой из них символ условной обработки && действует только на команду DIR, во второй — одновременно на две команды: DIR и COPY.

Примеры команд для работы с файловой системой

Рассмотрим некоторые наиболее часто используемые команды для работы с файловой системой. Отметим сначала несколько особенностей определения путей к файлам в Windows.

1.5 Основные команды

Полный список команд можно вывести набрав `HELP` в командной строке.

Команда CD

Текущий каталог можно изменить с помощью команды

CD [диск:][путь\]

Путь к требуемому каталогу указывается с учетом приведенных выше замечаний. Например, команда `CD \` выполняет переход в корневой каталог текущего диска. Если запустить команду CD без параметров, то на экран будут выведены имена текущего диска и каталога.

Команда COPY

Одной из наиболее часто повторяющихся задач при работе на компьютере является копирование и перемещение файлов из одного места в другое. Для копирования одного или нескольких файлов используется команда `COPY`.

Синтаксис этой команды:

COPY [/A|/B] источник [/A|/B] [+ источник [/A|/B] [+ ...]]
[результат [/A|/B]] [/V][/Y|/–Y]

Краткое описание параметров и ключей команды `COPY` приведено в табл. 1.1.

Таблица 1.1. Параметры и ключи команды COPY

Параметр	Описание
источник	Имя копируемого файла или файлов
/A	Файл является текстовым файлом ASCII, то есть конец файла обозначается символом с кодом ASCII 26 (<Ctrl>+<Z>)
/B	Файл является двоичным. Этот ключ указывает на то, что интерпретатор команд должен при копировании считывать из источника число байт, заданное размером в каталоге копируемого файла
результат	Каталог для размещения результата копирования и/или имя создаваемого

	файла
/V	Проверка правильности копирования путем сравнения файлов после копирования
/Y	Отключение режима запроса подтверждения на замену файлов
/-Y	Включение режима запроса подтверждения на замену файлов

Приведем примеры использования команды COPY.

Копирование файла abc.txt из текущего каталога в каталог D:\PROGRAM под тем же именем:

```
COPY abc.txt D:\PROGRAM
```

Копирование файла abc.txt из текущего каталога в каталог D:\PROGRAM под новым именем def.txt:

```
COPY abc.txt D:\PROGRAM\def.txt
```

Копирование всех файлов с расширением txt с диска A: в каталог 'Мои документы' на диске C:

```
COPY A:\*.txt "C:\Мои документы"
```

Если не задать в команде целевой файл, то команда COPY создаст копию файла-источника с тем же именем, датой и временем создания, что и исходный файл, и поместит новую копию в текущий каталог на текущем диске. Например, для того, чтобы скопировать все файлы из корневого каталога диска A: в текущий каталог, достаточно выполнить такую краткую команду:

```
COPY A:\*.*
```

В качестве источника или результата при копировании можно указывать имена не только файлов, но и устройств компьютера. Например, для того, чтобы распечатать файл abc.txt на принтере, можно воспользоваться командой копирования этого файла на устройство PRN:

```
COPY abc.txt PRN
```

Другой интересный пример: создадим новый текстовый файл и запишем в него информацию, без использования текстового редактора. Для этого достаточно ввести команду `COPY CON my.txt`, которая будет копировать то, что вы набираете на клавиатуре, в файл my.txt (если этот файл существовал, то он перезапишется, иначе — создастся). Для завершения ввода необходимо ввести символ конца файла, то есть нажать клавиши <Ctrl>+<Z>.

Команда COPY может также объединять (склеивать) нескольких файлов в один. Для этого необходимо указать единственный результирующий файл и несколько исходных. Это достигается путем использования групповых знаков (? и *) или формата файл1 + файл2 + файл3. Например, для объединения файлов 1.txt и 2.txt в файл 3.txt можно задать следующую команду:

```
COPY 1.txt+2.txt 3.txt
```

Объединение всех файлов с расширением dat из текущего каталога в один файл all.dat может быть произведено так:

```
COPY /B *.dat all.dat
```

Ключ /B здесь используется для предотвращения усечения соединяемых файлов, так как при комбинировании файлов команда COPY по умолчанию считает файлами текстовыми. Если имя целевого файла совпадает с именем одного из копируемых файлов (кроме первого), то исходное содержимое целевого файла теряется. Если имя целевого файла опущено, то в его качестве используется первый файл из списка. Например, команда `COPY 1.txt+2.txt` добавит к содержимому файла 1.txt содержимое файла 2.txt. Командой COPY можно воспользоваться и для присвоения какому-либо файлу текущей даты и времени без модификации его содержимого. Для этого нужно ввести команду типа

```
COPY /B 1.txt +,,
```

Здесь запятые указывают на пропуск параметра приемника, что и приводит к требуемому результату.

Команда COPY имеет и свои недостатки. Например, с ее помощью нельзя копировать скрытые и системные файлы, файлы нулевой длины, файлы из подкаталогов. Кроме того, если при копировании группы файлов COPY встретит файл, который в данный момент нельзя скопировать (например, он занят другим приложением), то процесс копирования полностью прервется, и остальные файлы не будут скопированы.

Команда XCOPY

Указанные при описании команды COPY проблемы можно решить с помощью команды XCOPY, которая предоставляет намного больше возможностей при копировании. Необходимо отметить, правда, что XCOPY может работать только с файлами и каталогами, но не с устройствами.

Синтаксис этой команды:

XCOPY источник [результат] [ключи]

Команда XCOPY имеет множество ключей, мы коснемся лишь некоторых из них. Ключ /D[:[дата]] позволяет копировать только файлы, измененные не ранее указанной даты. Если параметр дата не указан, то копирование будет производиться только если источник новее результата. Например, команда

```
XCOPY "C:\Мои документы\*.*" "D:\BACKUP\Мои документы" /D
```

скопирует в каталог 'D:\BACKUP\Мои документы' только те файлы из каталога 'C:\Мои документы', которые были изменены со времени последнего подобного копирования или которых вообще не было в 'D:\BACKUP\Мои документы'.

Ключ /S позволяет копировать все непустые подкаталоги в каталоге-источнике. С помощью же ключа /E можно копировать вообще все подкаталоги, включая и пустые. Если указан ключ /C, то копирование будет продолжаться даже в случае возникновения ошибок. Это бывает очень полезным при операциях копирования, производимых над группами файлов, например, при резервном копировании данных.

Ключ /I важен для случая, когда копируются несколько файлов, а файл назначения отсутствует. При задании этого ключа команда XCOPY считает, что файл назначения должен быть каталогом. Например, если задать ключ /I в команде копирования всех файлов с расширением txt из текущего каталога в несуществующий еще подкаталог TEXT,

```
XCOPY  *.txt TEXT /I
```

то подкаталог TEXT будет создан без дополнительных запросов.

Ключи /Q, /F и /L отвечают за режим отображения при копировании. При задании ключа /Q имена файлов при копировании не отображаются, ключа /F — отображаются полные пути источника и результата. Ключ /L обозначает, что отображаются только файлы, которые должны быть скопированы (при этом само копирование не производится).

С помощью ключа /H можно копировать скрытые и системные файлы, а с помощью ключа /R — заменять файлы с атрибутом "Только для чтения". Например, для копирования всех файлов из корневого каталога диска C: (включая системные и скрытые) в каталог SYS на диске D:, нужно ввести следующую команду:

```
XCOPY C:\*.* D:\SYS /H
```

Ключ /T позволяет применять XCOPY для копирования только структуры каталогов источника, без дублирования находящихся в этих каталогах файлов, причем пустые каталоги и подкаталоги не включаются. Для того, чтобы все же включить пустые каталоги и подкаталоги, нужно использовать комбинацию ключей /T /E.

Используя XCOPY можно при копировании обновлять только уже существующие файлы (новые файлы при этом не записываются). Для этого применяется ключ /U. Например, если в каталоге C:\2 находились файлы a.txt и b.txt, а в каталоге C:\1 — файлы a.txt, b.txt, c.txt и d.txt, то после выполнения команды

```
XCOPY C:\1 C:\2 /U
```

в каталоге C:\2 по-прежнему останутся лишь два файла a.txt и b.txt, содержимое которых будет заменено содержимым соответствующих файлов из каталога C:\1.Если с помощью

XCOPY копировался файл с атрибутом "Только для чтения", то по умолчанию у файла-копии этот атрибут снимется. Для того, чтобы копировать не только данные, но и полностью атрибуты файла, необходимо использовать ключ /K.
Ключи /Y и /-Y определяют, нужно ли запрашивать подтверждение перед заменой файлов при копировании. /Y означает, что такой запрос нужен, /-Y — не нужен.

Команда DIR

Еще одной очень полезной командой является DIR [диск:][путь][имя_файла] [ключи], которая используется для вывода информации о содержимом дисков и каталогов. Параметр [диск:][путь] задает диск и каталог, содержимое которого нужно вывести на экран. Параметр [имя_файла] задает файл или группу файлов, которые нужно включить в список. Например, команда

```
DIR C:\*.bat
```

выведет на экран все файлы с расширением bat в корневом каталоге диска C:. Если задать эту команду без параметров, то выводится метка диска и его серийный номер, имена (в коротком и длинном вариантах) файлов и подкаталогов, находящихся в текущем каталоге, а также дата и время их последней модификации. После этого выводится число файлов в каталоге, общий объем (в байтах), занимаемый файлами, и объем свободного пространства на диске. Например:

```
Том в устройстве C имеет метку PHYS1_PART2
 Серийный номер тома: 366D-6107
 Содержимое папки C:\aditor
.           <ПАПКА>     25.01.00  17:15 .
..          <ПАПКА>     25.01.00  17:15 ..
TEMPLT02 DAT         227  07.08.98   1:00 templt02.dat
UNINST1  000       1 093  02.03.99   8:36 UNINST1.000
HILITE   DAT       1 082  18.09.98  18:55 hilite.dat
TEMPLT01 DAT          48  07.08.98   1:00 templt01.dat
UNINST0  000      40 960  15.04.98   2:08 UNINST0.000
TTABLE   DAT         357  07.08.98   1:00 ttable.dat
ADITOR   EXE      461 312  01.12.99  23:13 aditor.exe
README   TXT        3 974  25.01.00  17:26 readme.txt
ADITOR   HLP       24 594  08.10.98  23:12 aditor.hlp
ТЕКСТО~1 TXT           0  11.03.01   9:02 Текстовый файл.txt
       11 файлов       533 647 байт
        2 папок     143 261 696 байт свободно
```

С помощью ключей команды DIR можно задать различные режимы расположения, фильтрации и сортировки. Например, при использовании ключа /W перечень файлов выводится в широком формате с максимально возможным числом имен файлов или каталогов на каждой строке. Например:

```
Том в устройстве C имеет метку PHYS1_PART2
 Серийный номер тома: 366D-6107
 Содержимое папки C:\aditor
[.]             [..]            TEMPLT02.DAT    UNINST1.000       HILITE.DAT
TEMPLT01.DAT    UNINST0.000       TTABLE.DAT      ADITOR.EXE
        README.TXT
ADITOR.HLP        ТЕКСТО~1.TXT
       11 файлов       533 647 байт
        2 папок     143 257 600 байт свободно
```

С помощью ключа /A[[:]атрибуты] можно вывести имена только тех каталогов и файлов, которые имеют заданные атрибуты (R — "Только чтение", A — "Архивный", S — "Системный", H — "Скрытый", префикс "–" имеет значение НЕ). Если ключ /A

используется более чем с одним значением атрибута, будут выведены имена только тех файлов, у которых все атрибуты совпадают с заданными. Например, для вывода имен всех файлов в корневом каталоге диска C:, которые одновременно являются скрытыми и системными, можно задать команду

DIR C:\ /A:HS

а для вывода всех файлов, кроме скрытых — команду

DIR C:\ /A:-H

Отметим здесь, что атрибуту каталога соответствует буква D, то есть для того, чтобы, например, вывести список всех каталогов диска C:, нужно задать команду

DIR C: /A:D

Ключ `/O[[:]сортировка]` задает порядок сортировки содержимого каталога при выводе его командой DI R. Если этот ключ опущен, DI R печатает имена файлов и каталогов в том порядке, в котором они содержатся в каталоге. Если ключ / O задан, а параметр сортировка не указан, то `DIR` выводит имена в алфавитном порядке. В параметре сортировка можно использовать следующие значения: `N` — по имени (алфавитная), `S` — по размеру (начиная с меньших), `E` — по расширению (алфавитная), `D` — по дате (начиная с более старых), `A` — по дате загрузки (начиная с более старых), `G` — начать список с каталогов. Префикс "–" означает обратный порядок. Если задается более одного значения порядка сортировки, файлы сортируются по первому критерию, затем по второму и т.д.

Ключ `/S` означает вывод списка файлов из заданного каталога и его подкаталогов.

Ключ `/B` перечисляет только названия каталогов и имена файлов (в длинном формате) по одному на строку, включая расширение. При этом выводится только основная информация, без итоговой. Например:

```
templt02.dat
UNINST1.000
hilite.dat
templt01.dat
UNINST0.000
ttable.dat
aditor.exe
readme.txt
aditor.hlp
Текстовый файл.txt
```

Команды MKDIR и RMDIR

Для создания нового каталога и удаления уже существующего пустого каталога используются команды `MKDIR [диск:]путь` и `RMDIR [диск:]путь [ключи]` соответственно (или их короткие аналоги `MD` и `RD`). Например:

```
MKDIR "C:\Примеры"
RMDIR "C:\Примеры"
```

Команда `MKDIR` не может быть выполнена, если каталог или файл с заданным именем уже существует. Команда `RMDIR` не будет выполнена, если удаляемый каталог не пустой.

Команда DEL

Удалить один или несколько файлов можно с помощью команды

DEL [диск:][путь]имя_файла [ключи]

Для удаления сразу нескольких файлов используются групповые знаки `?` и `*`. Ключ `/S` позволяет удалить указанные файлы из всех подкаталогов, ключ `/F` – принудительно удалить файлы, доступные только для чтения, ключ `/A[[:]атрибуты]` – отбирать файлы для удаления по атрибутам (аналогично ключу `/A[[:]атрибуты]` в команде `DIR`).

Команда REN

Переименовать файлы и каталоги можно с помощью команды `RENAME` (`REN`). Синтаксис этой команды имеет следующий вид:

REN [диск:][путь][каталог1|файл1] [каталог2|файл2]

Здесь параметр `каталог1|файл1` определяет название каталога/файла, которое нужно изменить, а `каталог2|файл2` задает новое название каталога/файла. В любом параметре команды `REN` можно использовать групповые символы ? и *. При этом представленные шаблонами символы в параметре `файл2` будут идентичны соответствующим символам в параметре `файл1`. Например, чтобы изменить у всех файлов с расширением txt в текущей директории расширение на doc, нужно ввести такую команду:

REN *.txt *.doc

Если файл с именем файл2 уже существует, то команда `REN` прекратит выполнение, и произойдет вывод сообщения, что файл уже существует или занят. Кроме того, в команде `REN` нельзя указать другой диск или каталог для создания результирующих каталога и файла. Для этой цели нужно использовать команду `MOVE`, предназначенную для переименования и перемещения файлов и каталогов.

Команда MOVE

Синтаксис команды для перемещения одного или более файлов имеет вид:

MOVE [/Y|/–Y] [диск:][путь]имя_файла1[,...] результирующий_файл

Синтаксис команды для переименования папки имеет вид:

MOVE [/Y|/–Y] [диск:][путь]каталог1 каталог2

Здесь параметр `результирующий_файл` задает новое размещение файла и может включать имя диска, двоеточие, имя каталога, либо их сочетание. Если перемещается только один файл, допускается указать его новое имя. Это позволяет сразу переместить и переименовать файл. Например,

MOVE "C:\Мои документы\список.txt" D:\list.txt

Если указан ключ `/-Y`, то при создании каталогов и замене файлов будет выдаваться запрос на подтверждение. Ключ `/Y` отменяет выдачу такого запроса.

1.6 Учебное задание

1. Запустить командный процессор: `cmd.exe`
2. Создать на диске E: папку: `mkdir test или md test`
3. Войти в созданную папку: `cd test`
4. В паке создать текстовый файл с информацией о студенте (ФИО, дата рождения, группа): `Copy con my.txt`
   Примечание: Для окончания ввода нажать CTRL+Z и затем ENTER.
5. Вывести содержимое паки на экран: `dir`
6. Вывести содержимое созданного файла на экран: `type my.txt`
7. В созданной папке создать две папки с именами A1 и A2.
   Примечание: Для повтора предыдущей команды удобно использовать кнопки – «вверх», «вниз», «вправо».
8. Скопировать *my.txt* в папку A1: `copy my.txt A1`.
9. В папке A2 создать файл с деревом каталога *test*: `tree . >A2\tree.txt`
10. Переименовать файл в папке A1 в *old.txt*: `ren a1\ my.txt old.txt`
11. С помощью команды *move* поменять содержимое папок A1 и A2.
12. Удалить все папки и файлы в исходной папке *test*: `rd /s .`

1.7 Контрольное задание

Задание выполняется в командном процессоре. Все команды привести в отчете по пунктам.

1. Создать папку ***Zadanie1***, а в ней три папки A1, A2, A3.
2. В папку A1 скопировать все файлы с расширением ***ini*** из каталога ***c:\windows***.
3. В папке A2 создать файл с именами файлов в паке ***c:\windows***.

4. скопировать содержимое папок A1 и A2 в A3 и вывести ее содержимое на экран. При выполнении задания использовать группировку команд (***&***).
5. Заменить расширение у всех файлов из папки A1 с ***ini*** на ***bak*** и перед именем каждого файла поставить символ ~.
6. Сохранить значения системных переменных окружения в файле с именем ***set.txt*** в папке A3.
7. Создать свою системную переменную myname с фамилией студента. Сохранить значения системных переменных окружения в файле с именем ***new_set.txt*** в папке A3. Файл привести в отчете.
8. Показать созданную структуру преподавателю .Удалить папки A1, A2, A3.

2. Язык интерпретатора Cmd.exe. Командные файлы

Язык оболочки командной строки (shell language) в Windows реализован в виде командных (или пакетных) файлов. Командный файл в Windows — это обычный текстовый файл с расширением bat или cmd, в котором записаны допустимые команды операционной системы (как внешние, так и внутренние), а также некоторые дополнительные инструкции и ключевые слова, придающие командным файлам некоторое сходство с алгоритмическими языками программирования.

2.1 Вывод сообщений и дублирование команд

По умолчанию команды пакетного файла перед исполнением выводятся на экран, что выглядит не очень эстетично. С помощью команды `ECHO OFF` можно отключить дублирование команд, идущих после нее (сама команда `ECHO OFF` при этом все же дублируется). Например,

```
REM Следующие две команды будут дублироваться на экране …
DIR C:\
ECHO OFF
REM А остальные уже не будут
DIR D:\
```

Для восстановления режима дублирования используется команда `ECHO ON`. Кроме этого, можно отключить дублирование любой отдельной строки в командном файле, написав в начале этой строки символ @ например:

```
ECHO ON
REM Команда DIR C:\ дублируется на экране
DIR C:\
REM А команда DIR D:\ — нет
@DIR D:\
```

Таким образом, если поставить в самое начало файла команду

```
@ECHO OFF
```

то это решит все проблемы с дублированием команд.

В пакетном файле можно выводить на экран строки с сообщениями. Делается это с помощью команды

```
ECHO сообщение
```

Например,

```
@ECHO OFF
ECHO Привет!
Команда ECHO. (точка должна следовать непосредственно за словом "ECHO")
выводит на экран пустую строку.
```

Часто бывает удобно для просмотра сообщений, выводимых из пакетного файла, предварительно полностью очистить экран командой `CLS`.

Используя механизм перенаправления ввода/вывода (символы > и >>), можно направить сообщения, выводимые командой `ECHO`, в определенный текстовый файл. Например:

```
@ECHO OFF
ECHO Привет! > hi.txt
ECHO Пока! >> hi.txt
С помощью такого метода можно, скажем,
 заполнять файлы-протоколы с отчетом о
  произведенных действиях. Например:
@ECHO OFF
REM Попытка копирования
XCOPY C:\PROGRAMS D:\PROGRAMS /s
```

```
REM Добавление сообщения в файл report.txt в случае
REM удачного завершения копирования
IF NOT ERRORLEVEL 1 ECHO Успешное копирование >> report.txt
```

2.2 Использование параметров командной строки

При запуске пакетных файлов в командной строке можно указывать произвольное число параметров, значения которых можно использовать внутри файла. Это позволяет, например, применять один и тот же командный файл для выполнения команд с различными параметрами.

Для доступа из командного файла к параметрам командной строки применяются символы `%0, %1, ..., %9` или `%*`. При этом вместо `%0` подставляется имя выполняемого пакетного файла, вместо `%1, %2, ..., %9` — значения первых девяти параметров командной строки соответственно, а вместо `%*` — все аргументы. Если в командной строке при вызове пакетного файла задано меньше девяти параметров, то "лишние" переменные из `%1 - %9` замещаются пустыми строками. Рассмотрим следующий пример. Пусть имеется командный файл copier.bat следующего содержания:

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

Если запустить его из командной строки с двумя параметрами, например

```
copier.bat C:\Programs D:\Backup
```

то на экран выведется сообщение

```
Файл copier.bat копирует каталог C:\Programs в D:\Backup
```

и произойдет копирование каталога C:\Programs со всеми его подкаталогами в D:\Backup.

При необходимости можно использовать более девяти параметров командной строки. Это достигается с помощью команды `SHIFT`, которая изменяет значения замещаемых параметров с `%0` по `%9`, копируя каждый параметр в предыдущий, то есть значение `%1` копируется в `%0`, значение `%2` – в `%1` и т.д. Замещаемому параметру `%9` присваивается значение параметра, следующего в командной строке за старым значением `%9`. Если же такой параметр не задан, то новое значение `%9` — пустая строка.

Рассмотрим пример. Пусть командный файл my.bat вызван из командной строки следующим образом:

```
my.bat p1 p2 p3
```

Тогда `%0=my.bat, %1=p1, %2=p2, %3=p3`, параметры `%4 - %9` являются пустыми строками. После выполнения команды `SHIFT` значения замещаемых параметров изменятся следующим образом: `%0=p1, %1=p2, %2=p3`, параметры `%3 - %9` – пустые строки.

При включении расширенной обработки команд `SHIFT` поддерживает ключ `/n`, задающий начало сдвига параметров с номера `n`, где `n` может быть числом от 0 до 9.

Например, в следующей команде:

```
SHIFT /2
```

параметр `%2` заменяется на `%3`, `%3` на `%4` и т.д., а параметры `%0` и `%1` остаются без изменений.

Команда, обратная `SHIFT` (обратный сдвиг), отсутствует. После выполнения `SHIFT` уже нельзя восстановить параметр `(%0)`, который был первым перед сдвигом. Если в командной строке задано больше десяти параметров, то команду `SHIFT` можно использовать несколько раз.

В командных файлах имеются некоторые возможности синтаксического анализа заменяемых параметров. Для параметра с номером `n (%n)` допустимы синтаксические конструкции (операторы), представленные в табл. 2.1.

Таблица 2.1. Операторы для заменяемых параметров	
Операторы	**Описание**
%~Fn	Переменная %n расширяется до полного имени файла
%~Dn	Из переменной %n выделяется только имя диска
%~Pn	Из переменной %n выделяется только путь к файлу
%~Nn	Из переменной %n выделяется только имя файла
%~Xn	Из переменной %n выделяется расширение имени файла
%~Sn	Значение операторов N и X для переменной %n изменяется так, что они работают с кратким именем файла
%~$PATH:n	Проводится поиск по каталогам, заданным в переменной среды PATH, и переменная %n заменяется на полное имя первого найденного файла. Если переменная PATH не определена или в результате поиска не найден ни один файл, эта конструкция заменяется на пустую строку. Естественно, здесь переменную PATH можно заменить на любое другое допустимое значение

Данные синтаксические конструкции можно объединять друг с другом, например:
%~DPn — из переменной %n выделяется имя диска и путь,
%~NXn — из переменной %n выделяется имя файла и расширение.
Рассмотрим следующий пример. Пусть мы находимся в каталоге C:\TEXT и запускаем пакетный файл с параметром Рассказ.doc (%1=Рассказ.doc). Тогда применение операторов, описанных в табл. 2.1, к параметру %1 даст следующие результаты:

```
%~F1=C:\TEXT\Рассказ.doc
%~D1=C:
%~P1=\TEXT\
%~N1=Рассказ
%~X1=.doc
%DP1=C:\TEXT\
%NX1=Рассказ.doc
```

2.3 Работа с переменными среды

Внутри командных файлов можно работать с так называемыми переменными среды (или переменными окружения), каждая из которых хранится в оперативной памяти, имеет свое уникальное имя, а ее значением является строка. Стандартные переменные среды автоматически инициализируются в процессе загрузки операционной системы. Такими переменными являются, например, WINDIR, которая определяет расположение каталога Windows, TEMP, которая определяет путь к каталогу для хранения временных файлов Windows или PATH, в которой хранится системный путь (путь поиска), то есть список каталогов, в которых система должна искать выполняемые файлы или файлы совместного доступа (например, динамические библиотеки). Кроме того, в командных файлах с помощью команды SET можно объявлять собственные переменные среды.
Получение значения переменной
Для получения значения определенной переменной среды нужно имя этой переменной заключить в символы %. Например:

```
@ECHO OFF
CLS
REM Создание переменной MyVar
SET MyVar=Привет
REM Изменение переменной
SET MyVar=%MyVar%!
```

```
ECHO Значение переменной MyVar: %MyVar%
REM Удаление переменной MyVar
SET MyVar=
ECHO Значение переменной WinDir: %WinDir%
```

При запуске такого командного файла на экран выведется строка

```
Значение переменной MyVar: Привет!
Значение переменной WinDir: C:\WINDOWS
```

2.4 Преобразования переменных как строк

С переменными среды в командных файлах можно производить некоторые манипуляции. Во-первых, над ними можно производить операцию конкатенации (склеивания). Для этого нужно в команде `SET` просто написать рядом значения соединяемых переменных. Например,

```
  SET A=Раз
SET B=Два
SET C=%A%%B%
```

После выполнения в файле этих команд значением переменной C будет являться строка `'РазДва'`. Не следует для конкатенации использовать знак `+`, так как он будет воспринят просто в качестве символа. Например, после запуска файл следующего содержания

```
  SET A=Раз
SET B=Два
SET C=A+B
ECHO Переменная C=%C%
SET D=%A%+%B%
ECHO Переменная D=%D%
```

на экран выведутся две строки:

```
Переменная C=A+B
Переменная D=Раз+Два
```

2.5 Операции с переменными как с числами

При включенной расширенной обработке команд (этот режим в Windows XP используется по умолчанию) имеется возможность рассматривать значения переменных среды как числа и производить с ними арифметические вычисления. Для этого используется команда `SET` с ключом `/A`. Приведем пример пакетного файла add.bat, складывающего два числа, заданных в качестве параметров командной строки, и выводящего полученную сумму на экран:

```
@ECHO OFF
REM В переменной M будет храниться сумма
SET /A M=%1+%2
ECHO Сумма %1 и %2 равна %M%
REM Удалим переменную M
SET M=
```

2.6 Локальные изменения переменных

Все изменения, производимые с помощью команды SET над переменными среды в командном файле, сохраняются и после завершения работы этого файла, но действуют только внутри текущего командного окна. Также имеется возможность локализовать изменения переменных среды внутри пакетного файла, то есть автоматически восстанавливать значения всех переменных в том виде, в каком они были до начала запуска этого файла. Для этого используются две команды: SETLOCAL и ENDLOCAL.

Команда SETLOCAL определяет начало области локальных установок переменных среды. Другими словами, изменения среды, внесенные после выполнения SETLOCAL, будут являться локальными относительно текущего пакетного файла. Каждая команда SETLOCAL должна иметь соответствующую команду ENDLOCAL для восстановления прежних значений переменных среды. Изменения среды, внесенные после выполнения команды ENDLOCAL, уже не являются локальными относительно текущего пакетного файла; их прежние значения не будут восстановлены по завершении выполнения этого файла.

2.7 Приостановка выполнения командных файлов

Для того, чтобы вручную прервать выполнение запущенного bat-файла, нужно нажать клавиши <Ctrl>+<C> или <Ctrl>+<Break>. Однако часто бывает необходимо программно приостановить выполнение командного файла в определенной строке с выдачей запроса на нажатие любой клавиши. Это делается с помощью команды PAUSE. Перед запуском этой команды полезно с помощью команды ECHO информировать пользователя о действиях, которые он должен произвести. Например:

```
ECHO Вставьте дискету в дисковод A: и нажмите любую клавишу
PAUSE
```

2.8 Операторы перехода

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

Пусть имеется командный файл следующего содержания:

```
@ECHO OFF
COPY %1 %2
GOTO Label1
ECHO Эта строка никогда не выполнится
:Label1
REM Продолжение выполнения
DIR %2
После того, как в этом файле мы доходим до команды
GOTO Label1
его выполнение продолжается со строки
REM Продолжение выполнения
```

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

Также для перехода к метке внутри текущего командного файла кроме команды GOTO можно использовать и рассмотренную выше команду CALL:

```
CALL :метка аргументы
```

При вызове такой команды создается новый контекст текущего пакетного файла с заданными аргументами, и управление передается на инструкцию, расположенную сразу после метки. Для выхода из такого пакетного файла необходимо два раза достичь его конца. Первый выход возвращает управление на инструкцию, расположенную сразу после строки CALL, а второй выход завершает выполнение пакетного файла. Например, если запустить с параметром "Копия-1" командный файл следующего содержания:

```
@ECHO OFF
ECHO %1
```

```
CALL :2 Копия-2
:2
ECHO %1
```
то на экран выведутся три строки:
```
Копия-1
Копия-2
Копия-1
```
Таким образом, подобное использование команды CALL очень похоже на обычный вызов подпрограмм (процедур) в алгоритмических языках программирования.

2.9 Операторы условия

С помощью команды I F … ELSE (ключевое слово ELSE может отсутствовать) в пакетных файлах можно выполнять обработку условий нескольких типов. При этом если заданное после IF условие принимает истинное значение, система выполняет следующую за условием команду (или несколько команд, заключенных в круглые скобки), в противном случае выполняется команда (или несколько команд в скобках), следующие за ключевым словом ELSE.

Проверка значения переменной

Первый тип условия используется обычно для проверки значения переменной. Для этого применяются два варианта синтаксиса команды IF:
```
IF [NOT] строка1==строка2 команда1 [ELSE команда2]
```
(квадратные скобки указывают на необязательность заключенных в них параметров) или
```
IF [/I] [NOT] строка1 оператор_сравнения строка2 команда
```
Рассмотрим сначала первый вариант. Условие строка1==строка2 (здесь необходимо писать именно два знака равенства) считается истинным при точном совпадении обеих строк. Параметр NOT указывает на то, что заданная команда выполняется лишь в том случае, когда сравниваемые строки не совпадают.

Строки могут быть литеральными или представлять собой значения переменных (например, %1 или %TEMP%). Кавычки для литеральных строк не требуются. Например,
```
IF %1==%2 ECHO Параметры совпадают!
IF %1==Петя ECHO Привет, Петя!
```
Отметим, что при сравнении строк, заданных переменными, следует проявлять определенную осторожность. Дело в том, что значение переменной может оказаться пустой строкой, и тогда может возникнуть ситуация, при которой выполнение командного файла аварийно завершится. Например, если вы не определили с помощью команды SET переменную MyVar, а в файле имеется условный оператор типа
```
IF %MyVar%==C:\ ECHO Ура!!!
```
то в процессе выполнения вместо %MyVar% подставится пустая строка и возникнет синтаксическая ошибка. Такая же ситуация может возникнуть, если одна из сравниваемых строк является значением параметра командной строки, так как этот параметр может быть не указан при запуске командного файла. Поэтому при сравнении строк нужно приписывать к ним в начале какой-нибудь символ, например:
```
IF -%MyVar%==-C:\ ECHO Ура!!!
```
С помощью команд IF и SHIFT можно в цикле обрабатывать все параметры командной строки файла, даже не зная заранее их количества. Например, следующий командный файл (назовем его primer.bat) выводит на экран имя запускаемого файла и все параметры командной строки:
```
@ECHO OFF
ECHO Выполняется файл: %0
ECHO.
ECHO Файл запущен со следующими параметрами...
```

```
REM Начало цикла
:BegLoop
IF -%1==- GOTO ExitLoop
ECHO %1
REM Сдвиг параметров
SHIFT
REM Переход на начало цикла
GOTO BegLoop
:ExitLoop
REM Выход из цикла
ECHO.
ECHO Все.
```

Если запустить primer.bat с четырьмя параметрами:

```
primer.bat А Б В Г
```

то в результате выполнения на экран выведется следующая информация:

```
Выполняется файл: primer.bat

Файл запущен со следующими параметрами:
А
Б
В
Г

Все.
```

Рассмотрим теперь оператор `IF` в следующем виде:

```
IF [/I] строка1 оператор_сравнения строка2 команда
```

Синтаксис и значение `операторов_сравнения` представлены в табл. 2.2.

Таблица 2.2. Операторы сравнения в IF

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

Приведем пример использования операторов сравнения:

```
@ECHO OFF
CLS
IF -%1 EQL –Вася ECHO Привет, Вася!
IF -%1 NEQ –Вася ECHO Привет, но Вы не Вася!
```

Ключ `/I`, если он указан, задает сравнение текстовых строк без учета регистра. Ключ `/I` можно также использовать и в форме `строка1==строка2` команды `IF`. Например, условие

```
IF /I DOS==dos …
```

будет истинным.

Проверка существования заданного файла

Второй способ использования команды `IF` — это проверка существования заданного файла. Синтаксис для этого случая имеет вид:

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

Условие считается истинным, если указанный файл существует. Кавычки для имени файла не требуются. Приведем пример командного файла, в котором с помощью такого варианта команды IF проверяется наличие файла, указанного в качестве параметра командной строки.

```
@ECHO OFF
IF -%1==- GOTO NoFileSpecified
IF NOT EXIST %1 GOTO FileNotExist

REM Вывод сообщения о найденном файле
ECHO Файл '%1' успешно найден.
GOTO :EOF

:NoFileSpecified
REM Файл запущен без параметров
ECHO В командной строке не указано имя файла.
GOTO :EOF

:FileNotExist
REM Параметр командной строки задан, но файл не найден
ECHO Файл '%1' не найден.
```

Проверка наличия переменной среды

Аналогично файлам команда `IF` позволяет проверить наличие в системе определенной переменной среды:

```
IF DEFINED переменная команда1 [ELSE команда2]
```

Здесь условие DEFI NED применяется подобно условию `EXISTS` наличия заданного файла, но принимает в качестве аргумента имя переменной среды и возвращает истинное значение, если эта переменная определена. Например:

```
@ECHO OFF
CLS
IF DEFINED MyVar GOTO :VarExists
ECHO Переменная MyVar не определена
GOTO :EOF
:VarExists
ECHO Переменная MyVar определена,
ECHO ее значение равно %MyVar%
```

Проверка кода завершения предыдущей команды

Еще один способ использования команды `IF` — это проверка кода завершения (кода выхода) предыдущей команды. Синтаксис для `IF` в этом случае имеет следующий вид:

```
IF [NOT] ERRORLEVEL число команда1 [ELSE команда2]
```

Здесь условие считается истинным, если последняя запущенная команда или программа завершилась с кодом возврата, равным либо превышающим указанное число.

Составим, например, командный файл, который бы копировал файл my.txt на диск C: без вывода на экран сообщений о копировании, а в случае возникновения какой-либо ошибки выдавал предупреждение:

```
@ECHO OFF
XCOPY my.txt C:\ > NUL
REM Проверка кода завершения копирования
IF ERRORLEVEL 1 GOTO ErrOccurred
ECHO Копирование выполнено без ошибок.
GOTO :EOF

:ErrOccurred
```

ECHO При выполнении команды XCOPY возникла ошибка!
В операторе IF ERRORLEVEL ... можно также применять операторы сравнения чисел, приведенные в табл. 2.2. Например:
IF ERRORLEVEL LEQ 1 GOTO Case1

2.10 Организация циклов

В командных файлах для организации циклов используются несколько разновидностей оператора FOR, которые обеспечивают следующие функции:
выполнение заданной команды для всех элементов указанного множества;
выполнение заданной команды для всех подходящих имен файлов;
выполнение заданной команды для всех подходящих имен каталогов;
выполнение заданной команды для определенного каталога, а также всех его подкаталогов;
получение последовательности чисел с заданными началом, концом и шагом приращения;
чтение и обработка строк из текстового файла;
обработка строк вывода определенной команды.

Цикл FOR … IN … DO …

Самый простой вариант синтаксиса команды FOR для командных файлов имеет следующий вид:
FOR %%переменная IN (множество)
DO команда [параметры]
Внимание
Перед названием переменной должны стоять именно два знака процента (%%), а не один, как это было при использовании команды FOR непосредственно из командной строки.
Сразу приведем пример. Если в командном файле заданы строки
@ECHO OFF
FOR %%i IN (Раз,Два,Три) DO ECHO %%i
то в результате его выполнения на экране будет напечатано следующее:
Раз
Два
Три
Параметр множество в команде FOR задает одну или более текстовых строк, разделенных запятыми, которые вы хотите обработать с помощью заданной команды. Скобки здесь обязательны. Параметр команда [параметры] задает команду, выполняемую для каждого элемента множества, при этом вложенность команд FOR на одной строке не допускается. Если в строке, входящей во множество, используется запятая, то значение этой строки нужно заключить в кавычки. Например, в результате выполнения файла с командами
@ECHO OFF
FOR %%i IN ("Раз,Два",Три) DO ECHO %%i
на экран будет выведено
Раз,Два
Три
Параметр %%переменная представляет подставляемую переменную (счетчик цикла), причем здесь могут использоваться только имена переменных, состоящие из одной буквы. При выполнении команда FOR заменяет подставляемую переменную текстом каждой строки в заданном множестве, пока команда, стоящая после ключевого слова DO, не обработает все такие строки.
Замечание.
Чтобы избежать путаницы с параметрами командного файла %0 — %9, для переменных следует использовать любые символы кроме 0 – 9.

Параметр множество в команде FOR может также представлять одну или несколько групп файлов. Например, чтобы вывести в файл список всех файлов с расширениями txt и prn, находящихся в каталоге C:\TEXT, без использования команды DIR, можно использовать командный файл следующего содержания:

```
@ECHO OFF
FOR %%f IN (C:\TEXT\*.txt C:\TEXT\*.prn) DO ECHO %%f >> list.txt
```

При таком использовании команды FOR процесс обработки продолжается, пока не обработаются все файлы (или группы файлов), указанные во множестве.

Цикл FOR /D … IN … DO …

Следующий вариант команды FOR реализуется с помощью ключа /D:

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

В случае, если набор содержит подстановочные знаки, то команда выполняется для всех подходящих имен каталогов, а не имен файлов. Скажем, выполнив следующий командный файл:

```
@ECHO OFF
CLS
FOR /D %%f IN (C:\*.*) DO ECHO %%f
```

мы получим список всех каталогов на диске C:, например:

```
C:\Arc
C:\CYR
C:\MSCAN
C:\NC
C:\Program Files
C:\TEMP
C:\TeX
C:\WINNT
```

Цикл FOR /R … IN … DO …

С помощью ключа / R можно задать рекурсию в команде: FOR:

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

В этом случае заданная команда выполняется для каталога [диск:] путь, а также для всех подкаталогов этого пути. Если после ключа R не указано имя каталога, то выполнение команды начинается с текущего каталога. Например, для распечатки всех файлов с расширением txt в текущем каталоге и всех его подкаталогах можно использовать следующий пакетный файл:

```
@ECHO OFF
CLS
FOR /R %%f IN (*.txt) DO PRINT %%f
```

Если вместо набора указана только точка (.), то команда проверяет все подкаталоги текущего каталога. Например, если мы находимся в каталоге C:\TEXT с двумя подкаталогами BOOKS и ARTICLES, то в результате выполнения файла:

```
@ECHO OFF
CLS
FOR /R %%f IN (.) DO ECHO %%f
```

на экран выведутся три строки:

```
C:\TEXT\.
C:\TEXT\BOOKS\.
C:\TEXT\ARTICLES\.
```

Цикл FOR /L … IN … DO …

Ключ / L позволяет реализовать с помощью команды FOR арифметический цикл, в этом случае синтаксис имеет следующий: вид:

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

Здесь заданная после ключевого слова IN тройка (начало,шаг,конец) раскрывается в последовательность чисел с заданными началом, концом и шагом приращения. Так, набор (1,1,5) раскрывается в (1 2 3 4 5), а набор (5,-1,1) заменяется на (5 4 3 2 1). Например, в результате выполнения следующего командного файла:

```
@ECHO OFF
CLS
FOR /L %%f IN (1,1,5) DO ECHO %%f
```

переменная цикла %%f пробежит значения от 1 до 5, и на экране напечатаются пять чисел:

```
1
2
3
4
5
```

Числа, получаемые в результате выполнения цикла FOR /L, можно использовать в арифметических вычислениях. Рассмотрим командный файл my.bat следующего содержания:

```
@ECHO OFF
CLS
FOR /L %%f IN (1,1,5) DO CALL :2 %%f
GOTO :EOF
:2
SET /A M=10*%1
ECHO 10*%1=%M%
```

В третьей строке в цикле происходит вызов нового контекста файла my.bat с текущим значением переменной цикла %%f в качестве параметра командной строки, причем управление передается на метку :2 (см. описание CALL в разделе "Изменения в командах перехода"). В шестой строке переменная цикла умножается на десять, и результат записывается в переменную M. Таким образом, в результате выполнения этого файла выведется следующая информация:

```
10*1=10
10*2=20
10*3=30
10*4=40
10*5=50
```

Цикл FOR /F ... IN ... DO ...

Самые мощные возможности (и одновременно самый запутанный синтаксис) имеет команда: FOR с ключом /F:

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

Здесь параметр набор содержит имена одного или нескольких файлов, которые по очереди открываются, читаются и обрабатываются. Обработка состоит в чтении файла, разбиении его на отдельные строки текста и выделении из каждой строки заданного числа подстрок. Затем найденная подстрока используется в качестве значения переменной при выполнении основного тела цикла (заданной команды).

По умолчанию ключ /F выделяет из каждой строки файла первое слово, очищенное от окружающих его пробелов. Пустые строки в файле пропускаются. Необязательный параметр "ключи" служит для переопределения заданных по умолчанию правил обработки строк. Ключи представляют собой заключенную в кавычки строку, содержащую приведенные в табл. 2.3 ключевые слова:

Таблица 2.3. Ключи в команде FOR /F

Ключ	Описание

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

При использовании ключа TOKENS=X,Y,M-N создаются дополнительные переменные. Формат M-N представляет собой диапазон подстрок с номерами от M до N. Если последний символ в строке TOKENS= является звездочкой, то создается дополнительная переменная, значением которой будет весь текст, оставшийся в строке после обработки последней подстроки.

Разберем применение этой команды на примере пакетного файла parser.bat, который производит разбор файла myfile.txt:

```
@ECHO OFF
IF NOT EXIST myfile.txt GOTO :NoFile
FOR /F "EOL=; TOKENS=2,3* DELIMS=, " %%i IN
  (myfile.txt) DO @ECHO %%i %%j %%k
GOTO :EOF
:NoFile
ECHO Не найден файл myfile.txt!
```

Здесь во второй строке производится проверка наличия файла myfile.txt; в случае отсутствия этого файла выводится предупреждающее сообщение. Команда FOR в третьей строке обрабатывает файл myfile.txt следующим образом:

Пропускаются все строки, которые начинаются с символа точки с запятой (EOL=;).

Вторая и третья подстроки из каждой строки передаются в тело цикла, причем подстроки разделяются пробелами (по умолчанию) и/или запятыми (DELIMS=,).

В теле цикла переменная %%i используется для второй подстроки, %%j — для третьей, а %%k получает все оставшиеся подстроки после третьей.

В нашем примере переменная %%i явно описана в инструкции FOR, а переменные %%j и %%k описываются неявно с помощью ключа TOKENS=. Например, если в файле myfile.txt были записаны следующие три строки:

```
ААА БББББ ВВВВ,ГГГГГ ДДДД
ЕЕЕЕЕ,ЖЖЖЖ ЗЗЗЗ
;КККК ЛЛЛЛЛЛ ММММММ
```

то в результате выполнения пакетного файла parser.bat на экран выведется следующее:

```
БББББ ВВВВ ГГГГГ ДДДД
ЖЖЖЖ ЗЗЗЗ
```

Замечание

Ключ TOKENS= позволяет извлечь из одной строки файла до 26 подстрок, поэтому запрещено использовать имена переменных, начинающиеся не с букв английского алфавита (a–z). Следует помнить, что имена переменных FOR являются глобальными, поэтому одновременно не может быть активно более 26 переменных.

Команда FOR /F также позволяет обработать отдельную строку. Для этого следует ввести нужную строку в кавычках вместо набора имен файлов в скобках. Строка будет обработана так, как будто она взята из файла. Например, файл следующего содержания:

```
@ECHO OFF
FOR /F "EOL=; TOKENS=2,3* DELIMS=, " %%i IN
  ("ААА БББББ ВВВВ,ГГГГГ ДДДД") DO @ECHO %%i %%j %%k
```

при своем выполнении напечатает

БББ ВВВВ ГГГГГ ДДДД
Вместо явного задания строки для разбора можно пользоваться переменными среды, например:
```
@ECHO OFF
SET M=AAA БББ ВВВВ,ГГГГГ ДДДД
FOR /F "EOL=; TOKENS=2,3* DELIMS=,
  " %%i IN ("%M%") DO @ECHO %%i %%j %%k
```
Наконец, команда `FOR /F` позволяет обработать строку вывода другой команды. Для этого следует вместо набора имен файлов в скобках ввести строку вызова команды в апострофах (не в кавычках!). Строка передается для выполнения интерпретатору команд cmd.exe, а вывод этой команды записывается в память и обрабатывается так, как будто строка вывода взята из файла. Например, следующий командный файл:
```
@ECHO OFF
CLS
ECHO Имена переменных среды:
ECHO.
FOR /F "DELIMS==" %%i IN ('SET') DO ECHO %%i
```
выведет перечень имен всех переменных среды, определенных в настоящее время в системе.
В цикле `FOR` допускается применение тех же синтаксических конструкций (операторов), что и для заменяемых параметров (табл. 2.4).

Таблица 2.4. Операторы для переменных команды FOR

Операторы	Описание
`%~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`	Значение операторов `N` и `X` для переменной `%i` изменяется так, что они работают с кратким именем файла

Замечание
Если планируется использовать расширения подстановки значений в команде `FOR`, то следует внимательно подбирать имена переменных, чтобы они не пересекались с обозначениями формата.
Например, если мы находимся в каталоге C:\Program Files\Far и запустим командный файл следующего содержания:
```
@ECHO OFF
CLS
FOR %%i IN (*.txt) DO ECHO %%~Fi
```
то на экран выведутся полные имена всех файлов
 с расширением txt:
```
C:\Program Files\Far\Contacts.txt
C:\Program Files\Far\FarFAQ.txt
C:\Program Files\Far\Far_Site.txt
C:\Program Files\Far\License.txt
C:\Program Files\Far\License.xUSSR.txt
C:\Program Files\Far\ReadMe.txt
C:\Program Files\Far\register.txt
C:\Program Files\Far\WhatsNew.txt
```

2.11 Циклы и связывание времени выполнения для переменных

Как и в рассмотренном выше примере с составными выражениями, при обработке переменных среды внутри цикла могут возникать труднообъяснимые ошибки, связанные с ранними связыванием переменных. Рассмотрим пример. Пусть имеется командный файл следующего содержания:

```
SET a=
FOR %%i IN (Раз,Два,Три) DO SET a=%a%%%i
ECHO a=%a%
```

В результате его выполнения на экран будет выведена строка `"a=Три"`, то есть фактически команда

```
FOR %%i IN (Раз,Два,Три) DO SET a=%a%%%i
```

равносильна команде

```
FOR %%i IN (Раз,Два,Три) DO SET a=%%i
```

Для исправления ситуации нужно, как и в случае с составными выражениями, вместо знаков процента (%) использовать восклицательные знаки и предварительно включить режим связывания времени выполнения командой `SETLOCAL ENABLEDELAYEDEXPANSION`. Таким образом, наш пример следует переписать следующим образом:

```
SETLOCAL ENABLEDELAYEDEXPANSION
SET a=
FOR %%i IN (Раз,Два,Три) DO SET a=!a!%%i
ECHO a=%a%
```

В этом случае на экран будет выведена строка `"a=РазДваТри"`.

2.12 Учебные задания

В каждом задании разработать пакетный файл. **Все параметры передаются через командную строку. Выполнить обязательную проверку на наличие всех требуемых параметров.** Командный файл и результаты его работы привести в отчете. *Обязательна демонстрация работы пакетного файла преподавателю.*

Пример

Выполнить резервное копирование содержимого заданной папки (с вложенными) в указанное место. Папки задать как параметры. Выполнить проверку на наличие заданных параметров.

```
@ECHO OFF
CLS
ECHO Файл %0 копирует каталог %1 в %2
IF -%1==- GOTO NoParam
IF -%2==- GOTO NoParam
XCOPY %1\ %2 /S
GOTO :eof
:NoParam
ECHO Не заданы необходимые параметры командной строки!
PAUSE
```

Задания

1. В заданном месте создать, если ее нет, папку с именем зарегистрированного пользователя. Создать и (или) дописать в файл с именем «Log In. log» время и дату входа в систему.ту входа в систему.

2. Выполнить резервное копирование файлов заданного расширения из указанной папки (с вложенными) в указанное место. Папки и расширение задать как параметры. Выполнить проверку на наличие заданных параметров, отключить вывод на экран списка копируемых файлов.

3. Удалить на заданном диске все файлы с заданным расширением (bak, tmp) вне зависимости от его атрибутов. Диск и расширение задать как параметры. Выполнить проверку на наличие заданных параметров.

4. Скопировать все файлы с заданным расширением в заданную папку без сохранения структуры каталогов (**четные номера компьютеров**), с сохранением структуры (**нечетные номера компьютеров**),. Папки и расширение задать как параметры. Продемонстрировать работу преподавателю.

5. Сохранить местоположение всех файлов с заданным расширением в файле с указанным именем. Расширение и имя файла задать как параметры. Заархивировать полученный список файлов архиватором rar (rar a имя_архива @файл_список)

6. Составить пакетный файл для копирования заданного файла в соответствии со списком папок хранящихся в другом файле. Имена файлов задать как параметры.

7. Создать на заданном диске папку с именем пользователя вошедшего в систему и сделать ее активной. Диск и имя файла задать как параметры.

8. Создать в указанном месте структуру папок, хранящуюся в заданном файле.

9. Выполнить резервное копирование с последующей архивацией файлов заданного расширения. Архиватор rar (rar a имя_архива @файл_список).

10. Разработать пакетный файл для построения системы студенческих каталогов с запросом на создание каталога группы и числа пользователей в группе.

11. Разработать пакетный файл для перезаписи файлов заданного расширения из одного каталога в другой с обновлением.

12. Разработать пакетный файл для копирования только новых и обновленных файлов заданного расширения из одного каталога в другой.

13. Разработать пакетный файл для вывода заданного сообщения заданному зарегистрированному пользователю. Сообщение и имя передаются как параметры.

14. Создать папку с именем зарегистрированного пользователя на заданном диске и выполнить копирование в ней всех файлов с заданным расширением и сохранением структуры папок.

15. Разработать пакетный файл для перехода в каталог с именем зарегистрированного пользователя, если он существует и архивирования его содержимого.