Давайте сначала дадим определение, что такое bat файл. Пакетные или batch файлы — это простые текстовые файлы, содержащие наборы команд (*инструкцию*) интерпретатора и имеющие расширение bat или cmd (*cmd работают только в ОС семейства NT*). Создавать и редактировать такие файлы можно при помощи обычного блокнота или любого другого текстового редактора.

Теперь Вы спросите, зачем нужно уметь писать такие бат файлы? И зачем они нужны? Попытаюсь объяснить.

Во-первых,  они используются для облегчения работы, т.е. например Вам нужно постоянно выполнять какую-нибудь операцию каждый день (*например, создавать архив тех или иных документов*), с помощью батника это можно автоматизировать, и Вы в этом уже не будете принимать участия.

Во-вторых, эти пакетные файлы очень мощные (*если конечно уметь их писать*), т.е. Вы можете даже написать неплохую программу (*я имею в виду по функционалу*). Лично мне они очень помогают в работе, и о некоторых вещах я уже просто забыл, когда это делал я в ручную.

Теперь перейдем непосредственно к основам этих батников. Как же они создаются, Вам нужно просто создать простой текстовый документ открыть его и сразу на вкладке «*Файл->сохранить как*», вписать вместо расширения «*Текстовый документ.txt*», например «*Текстовый документ.bat*» и сохранить, вот и получится у нас батник с расширением .bat, но он пока ничего не делает.

Для начала я приведу пример батника, который я использую у себя на работе для архивации документов.

"C:\Program Files\WinRAR\winrar.exe" a -r -dh -ed -agYYYY-mm-dd E:\arhaccounts\ d:\accounts\\*.doc

"C:\Program Files\WinRAR\winrar.exe" a -r -dh -ed -agYYYY-mm-dd E:\arhaccounts\ d:\accounts\\*.xls

"C:\Program Files\WinRAR\winrar.exe" a -r -dh -ed -agYYYY-mm-dd E:\arhaccounts\ d:\accounts\\*.txt

Теперь немного расскажу, что делает этот батник. Запускается WinRar, потом идут команды винрара:

* a — это добавить в архив;
* -r — обрабатывать вложенные папки;
* -dh — открывать совместно используемые файлы;
* -ed — не добавлять пустые папки;
* YYYY-mm-dd — добавляем к имени архива текущую дату (*формат даты*);
* E:\arhaccounts\ — путь, где будет лежать конечный архив;
* d:\accounts\\*.doc — путь и маска файлов, какие нужно заархивировать.

В данном случае мы архивируем все документы Word, Excel и текстовые файлы, архивировать остальные нам не требуется. У нас архивация идет на другой диск, и мы еще копируем полученный архив на другой компьютер, для того чтобы архивы хранились и в другом кабинете. Копирование идет по сети соответственно компьютер, на который копируется архив, должен быть включен. Для этого можно использовать такую команду:

copy E:\arhaccounts\\*.rar \\namecomp\arhiv\

**Примеры команд для bat файлов**

Теперь рассмотрим основные команды, которые Вы можете использовать.

Если Вам нужно удалить файл пропишите следующие:

Del d:\file\test.doc

Для того чтобы удалить весь каталог пропишите:

rd d:\file\

Вдруг Вам нужно удалять каждый раз все из какого-нибудь каталога, то используйте вот что:

echo Y| del d:\file\

где,

* del d:\file\ — это как раз и есть удаление всех файлов;
* echo Y| — команда подтверждает удаление т.к. если не прописать эту команду, то Вы увидите сообщение о подтверждение удаления – «Продолжить [Y(да)/N(нет)]», и Вам нужно будет каждый раз отвечать на этот вопрос.

Сейчас давайте рассмотрим пример посложней, в котором уже выполняется условие:

@echo off

"C:\Program Files\WinRAR\winrar.exe" x -O+ -IBCK d:\test\test.rar d:\test

IF not EXIST d:\test\123.rar GOTO 1

IF EXIST d:\test\123.rar GOTO 2

:2

"C:\Program Files\WinRAR\winrar.exe" x -O+ -IBCK d:\test\123.rar c:\

del d:\test\123.rar

:1

del d:\test\test.rar

end

Теперь объясняю, допустим, Вам нужно разархивировать архив test.rar, в котором будет много файлов, но если там окажется файл 123.rar его нужно будет разархивировать в корень диска C, а остальные файлы чтобы остались в том же каталоге не тронутыми.

По порядку, команда @echo off нужна, для того чтобы на экране у нас ничего не отражалось (*в принципе, если не нужно можете не писать эту строку*). Далее у нас запускается винрар и распаковывает архив test.rar в папку test. Затем идет условие, если в папке test (*после распаковки test.rar*) у нас не окажется файла 123.rar, то у нас просто выполнение батника переходит к строке :1 и потом просто удаляется файл test.rar за не надобностью т.к. все что нам нужно мы уже распаковали. Но если там оказывается файл 123.rar, то выполнение батника переходит к строке :2, после которой уже идет распаковка файла 123.rar в корень диска C. Другими словами у нас выполняется условие, если есть файл то делать это, если нет файла, делать вот это. Допустим, если не прописать условие в данном примере, то у нас батник выдаст ошибку, тогда когда у нас не окажется файла 123.rar в данной папке.

А сейчас рассмотрим такой пример, допустим, Вам нужно каждый раз перемещать из каталога расположенного на диске D файлы на флешку. Вы каждый раз должны будете заходить мой компьютер диск D, выбирать нужную папку, выделять все файлы из нее и вырезать, а потом только заходить на флешку и вставлять. С помощью батника это делается в один клик (*с одним условием, что флешка каждый раз будет, например диском G или тем какой у Вас*). Вот пример такого батника:

move "D:\catalog\\*.doc" G:\catalognaflehe\

И все файлы с расширением doc, которые находятся в каталоге D:\catalog будут перемещаться на флешку. Теперь хочу сказать, что в батниках можно использовать сценарии (*скрипты*) при помощи [Windows Scripting Host](https://info-comp.ru/programmirovanie/255-jscript-wsh.html" \o "Основы языка JScript - написание WSH скриптов) и если нужно, например чтобы выводилось сообщение, после того как файлы скопированы (*предыдущий пример*) вставьте вот это:

echo var WSHShell = WScript.CreateObject("WScript.Shell"); > %temp%\mes.js

echo WSHShell.Popup("Файлы Скопированы"); >> %temp%\mes.js

start %temp%\mes.js

deltree /y %temp%\mes.js

На самом деле про написание батников можно разговаривать много и в одну статью это, конечно же, не уместить, здесь я показал всего лишь принципы, которые используются при написании bat файлов, так сказать основу. Если Вы хотите узнать больше команд для написания батников, Вы можете легко посмотреть их, набрав в командной строке (*Пуск — Выполнить – cmd*) команду help, но там, конечно же, не все команды, что можно использовать в батниках. Удачи в написание BAT файлов (*батников*).

**Пример 1 – удаление старых архивов**

При архивировании чего-либо, многих интересует вопрос «***Как удалять старые архивы за ненадобностью с помощью батника?***». Например, все они лежат в одной папке и нужно удалять все архивы, которые старше 14 дней. После того как я почитал мануалы, полазил в Интернете, я могу предложить следующий способ.

Можно сделать так, что в папке с архивами будут храниться только определенное количество архивов, соответственно последних (*т.е. как раз в нашем случае за последние 2 недели*).

Это делается с помощью двух команд. Первая, это DIR, т.е. просто считываем все файлы в одной папке, и записываем их название в текстовый файл.

**dir D:\arhiv\\*.rar /a:-D/b/o:-D > list\_of\_files.txt**

где,

* **dir D:\arhiv\\*.rar** – это означает, что мы считываем все rar архивы в папке D:\arhiv\;
* **/a:-D** – это означает, что будут выводиться все файлы с указанными атрибутами, ключ -D означает, что нам нужны только файлы, не каталоги, префикс «-» как раз имеет значение отрицание, т.е. не каталоги, если бы мы написали просто D, то он считывал бы и каталоги;
* **/b** – это вывод только имен файлов;
* **/o:-D** – это сортировка, ключ –D означает, что будет выполнена сортировка по дате, но сначала более старые, для исправления этого мы уже знаем, что нам поможет префикс «-»;
* **>** — означает, что будет выполнено перенаправление вывода в файл list\_of\_files.txt, назвать можете по-другому.

Вот мы считали все наши архивы и записали их в файл, далее нам нужно перебрать все эти файлы и оставить только 14 штук, т.е. за последние 2 недели. Это делаем с помощью команды **FOR**, это своего рода цикл, который выполняет определенное действие для каждого файла в папке или каждой строки в файле, как в нашем случае.

**for /F «skip=13» %%i in (list\_of\_files.txt) do (del /Q «%%i»)**

где,

* **for** – сама команда для переборки;
* **/F «skip=13»** – это ключ с параметром, который означает, что не нужно обрабатывать первые 13 файлов, т.е. мы их пропускаем. Почему 13, а не 14 да потому что 14 архив (*т.е. сегодняшний, который должен создаться при выполнении этого батника*) еще не создался, поэтому и 13;
* **%%i** – переменная, в которой хранится название текущего файла;
* **In (list\_of\_files.txt)** – означает, что перебирать все строки именно в этом файле;
* **do (del /Q «%%i»)** —  говорит, что необходимо сделать с каждым, в нашем случае мы просто удаляем эти файлы с помощью del /Q ключ /Q, для того чтобы у нас не спрашивали подтверждения перед удалением. Советую для тестов del /Q заменить на echo, т.е. просто вывести на экран те файлы.

Итого у нас получился вот такой батник:

**dir D:\arhiv\\*.rar /a:-D/b/o:-D > list\_of\_files.txt**

**for /F «skip=13» %%i in (list\_of\_files.txt) do (del /Q «%%i»)**

Соответственно после этих строк можете писать сам код архивации, и в итоге у нас получится то, что в нашей папке будет храниться только 14 архивов, конечно же, самых последних.

**Пример 2 – использование переменных**

В батниках можно даже использовать переменные, как в настоящем языке программирования. Рассмотрим простейший пример использования переменных, например, мы хотим умножать на 2 число, которое мы введем в поле при запуске батника.

**@echo off**

**SET a=2**

**SET /p b=[введите второе число для умножения]**

**SET /a c=%a%\*%b%**

**echo %c%**

**pause**

Как вы поняли, переменные задаются с помощью команды SET. Для того чтобы использовать переменную в дальнейшем мы с обеих сторон переменной подставляем знак процента (%), для того чтобы командная строка поняла, что это именно переменная.

* **@echo off** – для того чтобы наши команды не отображались на экране;
* **SET a=2** – это просто задаем переменной «a» значение;
* **SET /p b=[введите второе число для умножения]** – это мы задаем переменной «b» значение, которое мы введем в поле, для того чтобы батник понял, что мы хотим сами ввести значение переменной, используется ключ /p;
* **SET /a c=%a%\*%b%** — задаем переменной «c» результат нашего выражение (*в нашем примере это умножение*);
* **echo %c%** — выводим на экран значение переменной «c»;
* **pause** – это мы просто ставим на паузу выполнение нашего bat файла, чтобы просто на всего увидеть все результаты.

Кстати, для того чтобы у Вас русские буквы нормально отображались в командной строке, сохраните бат файл в кодировке DOS-866.

С переменными разобрались, теперь давайте, применим это к нашему первому примеру, допустим, мы хотим оставлять не 14 архивов, а то количество, которое сами захотим, для этого мы с Вами при запуске батника будем вводить число архивов, которое нужно сохранить. Получится примерно вот так:

**@echo off**

**SET /p chislo=[сколько архивов сохранить?]**

**dir D:\test\\*.rar /a:-D/b/o:-D > list\_of\_files.txt**

**for /F «skip=%chislo%» %%i in (list\_of\_files.txt) do (del /Q «%%i»)**

Ну вот, примерно так, конечно на практике может это и не понадобится, но зато мы узнали, как можно использовать переменные.

Про переменные хочу еще сказать, что есть такие системные переменные как:

**%DATE%** — показывает текущую дату.

**%TIME%** — показывает текущее время.

Для примера выполните следующий код:

**echo %DATE%**

**echo %time:~0,-3%**

**pause**

Переменную %TIME% я написал именно таким образом, для того чтобы результат вывелся в более читабельном виде, попробуйте написать %TIME% и % TIME:~0,-3% у Вас, во втором случае уберутся 3 последних знака.

На самом деле системных переменных больше, просто эти могут потребоваться чаще остальных.

**Пример 3 – оператор условного выполнения IF**

Как и в других полноценных языках в батниках можно использовать оператор условного выполнения IF. Приведем небольшой пример, батник просто проверяет, существует файл или нет:

**@echo off**

**IF EXIST test.txt (**

**echo Файл существует**

**) ELSE (**

**echo Файла такого нет**

**)**

**Pause**

где,

**IF EXIST test.txt** – это как раз и идет проверка на наличие файла.

После, в скобках, идет то, что мы хотим сделать, если файл существует, а если файла не существует, то после ELSE, идет то, что нужно выполнить при отсутствии файла.

Теперь немного модифицируем наш пример с умножением на 2 введенное нами число, просто, если вдруг мы введем ноль, мы выведем соответствующие сообщение и попросим ввести заново число.

**@echo off**

**SET a=2**

**:metka**

**SET /p b=[введите второе число для умножения]**

**SET /a c=%a%\*%b%**

**if %c%==0 (echo вы ввели число 0, введите другое) else echo %c%**

**if %c%==0 (goto:metka)**

**pause**

Тут уже все знакомо, единственное, что при сравнении переменной «c» используется оператор сравнения == (*два равно*), так как просто равно (=) — это оператор присваивания. Если Вы заметили я здесь использовал оператор goto, т.е. переход к нужной метки. Другими словами, ставим метку и в зависимости от результата проверки условия, будет осуществлен переход к нужной метке.

Теперь хотелось бы отметить то, что многие используют в работе, например, для создания архива, программу winrar и, конечно же, используют ее в своих батниках, но многие задают вопросы по ключам, которые относятся к winrar. Вы не путайте ключи winrar, они используются только для этой программы, а не для всего, что есть в батниках, т.е. командной строке, например, если Вы используете 7zip, то ключи уже будут другие. Что касается ключей winrar то, полный и лучший справочник, на мой взгляд, есть, конечно же, в самом winrare. Для того чтобы посмотреть описание ключей winrar, откройте программу winrar зайдите в меню Справка, затем нажмите «*Содержание*», а далее выберите в самом содержание строку «*Режим командной строки*», где и будет описание всех ключей, даже приводятся простенькие примеры. Соответственно если у Вас английская версия winrar, то смысл такой же, только будет все на английском языке.

**3. Краткие сведения о командных файлах**

Командный файл состоит из последовательности строк, в каждой из которых может находиться либо вызов программы, либо вспомогательные команды. Большинство строк командного файла обрабатываются DOS так же, как если бы они вводились пользователем в командной строке. Вспомогательные команды служат для управления ходом работы командного файла. Наиболее распространенные вспомогательные команды:

**ЕСНО**

При формате вызова **ECHO OFF** отменяет вывод строк командного файла на экран при выполнении. Команда **ECHO ON** возобновляет их вывод на экран. Большинство командных файлов начинаются со строки**@ECHO OFF**. Символ "**@**" служит для отмены вывода на экран строки, следующей непосредственно за ним.

В остальных случаях команда **ЕСНО**выводит на экран текст, следующий за ней в строке.

**GOTO**

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

**IF**

Служит для проверки условия во время выполнения командного файла.

Формат команды: **IF *УСЛОВИЕ КОМАНДА*** .

Команда будет выполнена в том случае, если условие будет истинно. Однако, допуст м формат **IF NOT *УСЛОВИЕ КОМАНДА***. При этом команда выполнится, если условие ложно. В качестве команды часто используется **GOTO**.

В качестве условия могут быть использованы выражения:

**EXIST *ФАЙЛ***- истинно в том случае, если указанный файл существует.

***СТРОКА1 = = СТРОКА2***- истинно, если строки совпадают. Обычно в одну из строк входят *параметры командного файла*.

**PAUSE**

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

**REM**

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

**FOR**

Формат команды: **FOR %%x IN (*СПИСОК*) DO *КОМАНДА***

При выполнении *КОМАНДА*будет выполнена для каждого файла из списка. Эту команду можно использовать, если программа, которую нужно вызвать, не поддерживает "джокеров" - символов "\*" и "?" в именах файлов.

Пример**: FOR %%A IN (\*.TXT) DO MORE<%%A**

Эту команду можно использовать и в командной строке, но при этом вместо **%%х** нужно писать **%х**.

Параметры командного файла - слова, идущие при его вызове в командной строке вслед за именем самого файла.

Пример **- compare.bat example.txt.** **example2.txt**

В командном файле параметры указываются с помощью выражений **%х**, где х - цифра от 1 до 9. **%0** означает имя самого командного файла. Так, если в приведенном выше примере файл **compare.bat** содержит строку **type %1** , то во время выполнения она заменится на type **example.txt**.

Некоторые команды MS-DOS, необходимые в написании командных файлов:

Смена текущего каталога осуществляется командой

**CD *имя\_каталога***

Создание каталога осуществляется командой

**MD *имя\_каталога*.**

Копирование файла - командой

**COPY *имя\_файла1 имя\_файла2*.**

Удаление файла - командой

**DEL *имя\_файла*.**

Удаление каталога – командой

**DELTREE *имя\_каталога***

Файлы объединяются командой

**COPY *файл1* + *файл2 файл3***

после выполнения которой *файл3* содержит данные, находившиеся в файлах *файл1* и *файл2*.

**4. Практическая часть**

**4.1 Тексты программ**

1. **Файл BATCH1.BAT**

@echo offecho Копирование и удаление файла a:mkdir a:dir1mkdir a:dir1dir2copy c:alphaetagamma ext1.txt a:dir1dir2 extnew.txt

del c:alphaetagamma ext1.txt

echo Файл скопирован и удален

pause

1. **Файл SUMMA.BAT**

@echo offecho Объединение и переименование файлов copy c:d1anew.pas + c:d1new.pas c:d2cnew.pastype c:d2cnew.paspause

rename c:d1anew.pas aold.pas

rename c:d1new.pas bold.pas

echo Задание выполнено

1. **Файл \_EXIST.BAT**

@echo offecho Копирование файла в случае его отсутствия на дискете if not exist a:simp.for goto met1echo Файл simp.for уже есть на дискете

goto met2

:met1

copy c:forsimp.for a:simp.for

echo Файл simp.for скопирован на диск A:

:met2

1. **Файл BATCH4.BAT**

@echo offgoto m%1goto qend:m1a:mkdir a:mydircopy c:\*.com a:mydircopy c:\*.exe a:mydir

goto qend

:m2

a:

cd a:mydir

mkdir a:mydir ewdir

copy c:\*.com a:mydir ewdir

goto qend

:m3

dir a:mydir

goto qend

:m4

deltree a:mydir ewdir

goto qend

:met5

deltree a:mydir

:qend

* 1. **Протокол работы программ**

1. Файл BATCH1.BAT

Копирование и удаление файла

1 файл(а,ов) скопировано

Файл скопирован и удален

Нажмите любую клавишу . . .

1. Файл SUMMA.BAT

Объединение и переименование файлов

c:d1anew.pas

c:d1new.pas

1 файл(а,ов) скопировано

{ Содержимое файла anew.pas }

{ Содержимое файла bnew.pas }

Нажмите любую клавишу . . .Задание выполнено

1. Файл \_EXIST.BAT

а) файл отсутствует на дискете:Копирование файла в случае его отсутствия на дискете

1 файл(а,ов) скопировано

Файл simp.for скопирован на диск A:

б) файл есть на дискете:Копирование файла в случае его отсутствия на дискете

Файл simp.for уже есть на дискете

4. Файл BATCH4.BAT

а) запуск с параметром 1

{ список из 10 .com файлов }

10 файл(а,ов) скопировано

{ список из 8 .exe файлов }

8 файл(а,ов) скопировано

б) запуск с параметром 2

{ список из 10 .com файлов }

10 файл(а,ов) скопировано

в) запуск с параметром 3

{ вывод содержимого каталога A:MYDIR }

г) запуск с параметром 4

Удалить каталог “a:mydir ewdir” вместе со всеми подкаталогами? [Y(да)/N(нет)] y

Удаление a:mydir ewdir...

д) запуск с параметром 5

Удалить каталог “a:mydir” вместе со всеми подкаталогами? [Y(да)/N(нет)] y

Удаление a:mydir...

1. **Заключение**

Выполненные задания позволяют сделать вывод о целесообразности применения командных файлов в случае необходимости использования часто повторяющихся действий. Командные файлы являются наиболее удачным решением для реализации простых алгоритмов, связанных с операциями над файлами. Однако для реализации сложных алгоритмов язык командных файлов может оказаться недостаточным.