

# BATCH SCRIPTING

**tutorialspoint**  
SIMPLY EASY LEARNING

[www.tutorialspoint.com](http://www.tutorialspoint.com)



<https://www.facebook.com/tutorialspointindia>



<https://twitter.com/tutorialspoint>

# Batch Script - Quick Guide

Advertisements

## Batch Script - Overview

Batch Script is incorporated to automate command sequences which are repetitive in nature. Scripting is a way by which one can alleviate this necessity by automating these command sequences in order to make one's life at the shell easier and more productive. In most organizations, Batch Script is incorporated in some way or the other to automate stuff.

Some of the features of Batch Script are –

- Can read inputs from users so that it can be processed further.
- Has control structures such as for, if, while, switch for better automating and scripting.
- Supports advanced features such as Functions and Arrays.
- Supports regular expressions.
- Can include other programming codes such as Perl.

Some of the common uses of Batch Script are –

- Setting up servers for different purposes.
- Automating housekeeping activities such as deleting unwanted files or log files.
- Automating the deployment of applications from one environment to another.
- Installing programs on various machines at once.

Batch scripts are stored in simple text files containing lines with commands that get executed in sequence, one after the other. These files have the special extension BAT or CMD. Files of this type are recognized and executed through an interface (sometimes called a shell) provided by a system file called the command interpreter. On Windows systems, this interpreter is known as cmd.exe.

Running a batch file is a simple matter of just clicking on it. Batch files can also be run in a command prompt or the Start-Run line. In such case, the full path name must be used unless the file's path is in the path environment. Following is a simple example of a Batch

Script. This Batch Script when run deletes all files in the current directory.

```
:: Deletes All files in the Current Directory With Prompts and Warnings
::(Hidden, System, and Read-Only Files are Not Affected)
:: @ECHO OFF
DEL . DR
```

## Batch Script - Environment

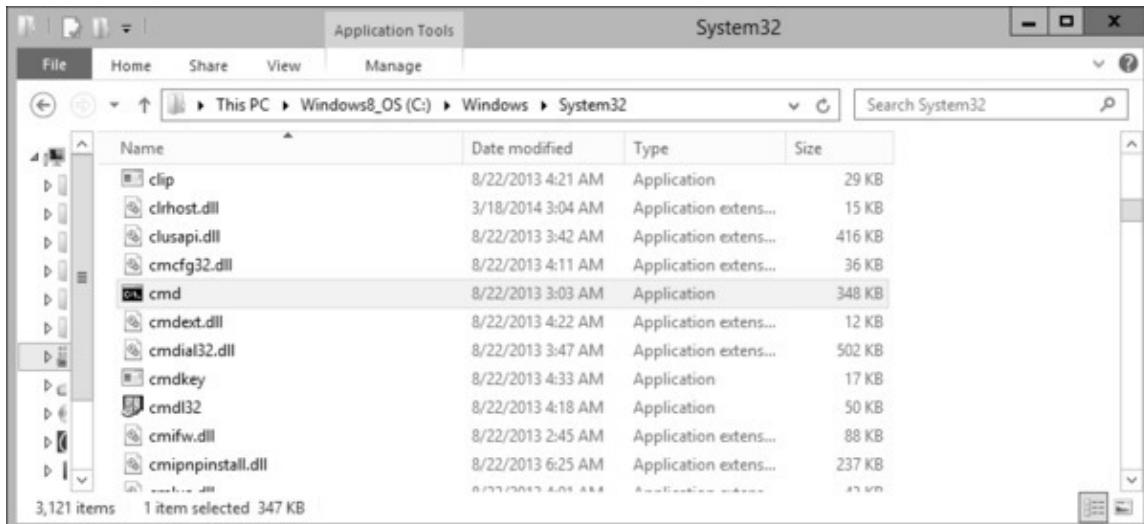
This chapter explains the environment related to Batch Script.

### Writing and Executing

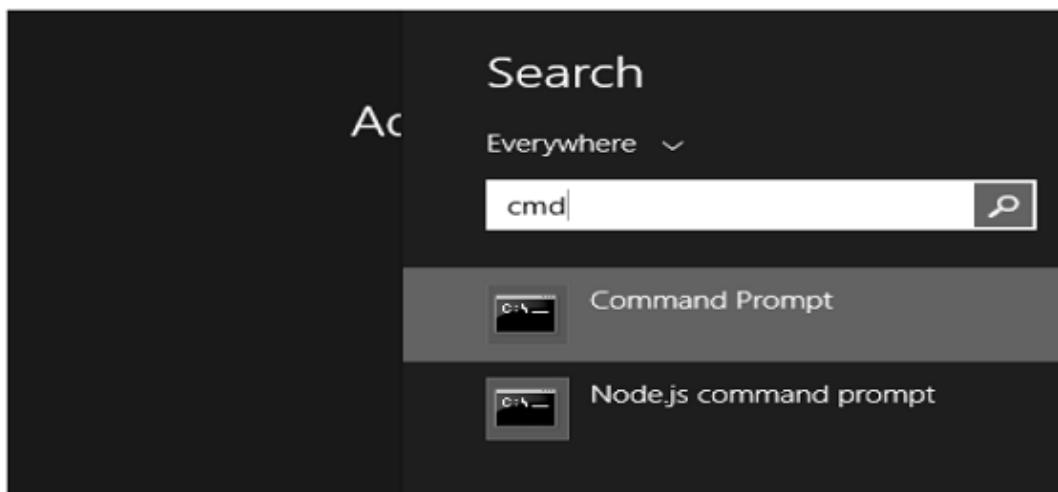
Typically, to create a batch file, notepad is used. This is the simplest tool for creation of batch files. Next is the execution environment for the batch scripts. On Windows systems, this is done via the command prompt or cmd.exe. All batch files are run in this environment.

Following are the different ways to launch cmd.exe –

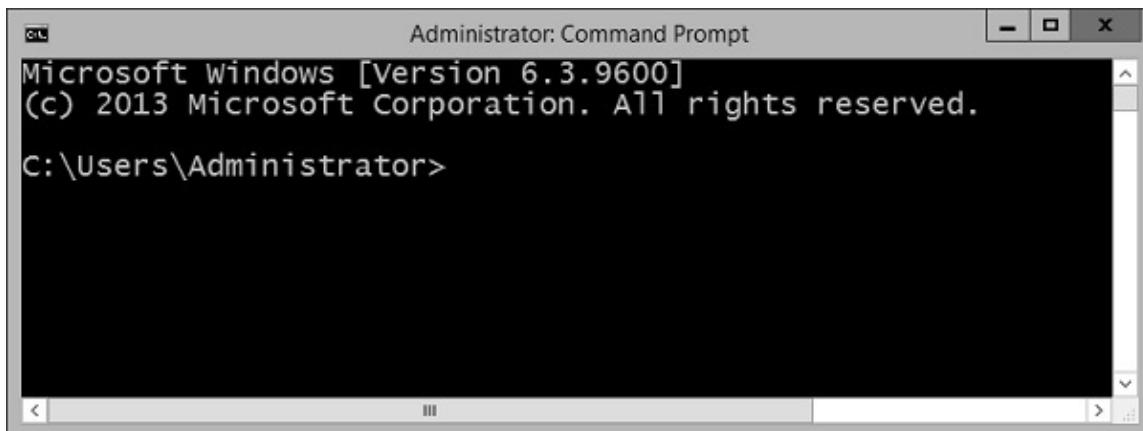
**Method 1** – Go to C:\Windows\System32 and double click on the cmd file.



**Method 2** – Via the run command – The following snapshot shows to find the command prompt(cmd.exe) on Windows server 2012.



Once the cmd.exe is launched, you will be presented with the following screen. This will be your environment for executing your batch scripts.



## Environment Variables

In order to run batch files from the command prompt, you either need to go to the location to where the batch file is stored or alternatively you can enter the file location in the path environment variable. Thus assuming that the batch file is stored in the location C:\Application\bin, you would need to follow these instructions for the PATH variable inclusion.

OS	Output
Windows	Append the String; C:\Application\bin to the end of the system variable PATH.

## Batch Script - Commands

In this chapter, we will look at some of the frequently used batch commands.

S.No	Commands & Description
1	<b>VER</b> This batch command shows the version of MS-DOS you are using.
2	<b>ASSOC</b> This is a batch command that associates an extension with a file type (FTYPE), displays existing associations, or deletes an association.
3	<b>CD</b> This batch command helps in making changes to a different directory, or displays

	the current directory.
4	<b>CLS</b> This batch command clears the screen.
5	<b>COPY</b> This batch command is used for copying files from one location to the other.
6	<b>DEL</b> This batch command deletes files and not directories.
7	<b>DIR</b> This batch command lists the contents of a directory.
8	<b>DATE</b> This batch command help to find the system date.
9	<b>ECHO</b> This batch command displays messages, or turns command echoing on or off.
10	<b>EXIT</b> This batch command exits the DOS console.
11	<b>MD</b> This batch command creates a new directory in the current location.
12	<b>MOVE</b> This batch command moves files or directories between directories.
13	<b>PATH</b> This batch command displays or sets the path variable.
14	<b>PAUSE</b> This batch command prompts the user and waits for a line of input to be entered.
15	<b>PROMPT</b> This batch command can be used to change or reset the cmd.exe prompt.
16	<b>RD</b> This batch command removes directories, but the directories need to be empty before they can be removed.

17	<b>REN</b> Renames files and directories
18	<b>REM</b> This batch command is used for remarks in batch files, preventing the content of the remark from being executed.
19	<b>START</b> This batch command starts a program in new window, or opens a document.
20	<b>TIME</b> This batch command sets or displays the time.
21	<b>TYPE</b> This batch command prints the content of a file or files to the output.
22	<b>VOL</b> This batch command displays the volume labels.
23	<b>ATTRIB</b> Displays or sets the attributes of the files in the current directory
24	<b>CHKDSK</b> This batch command checks the disk for any problems.
25	<b>CHOICE</b> This batch command provides a list of options to the user.
26	<b>CMD</b> This batch command invokes another instance of command prompt.
27	<b>COMP</b> This batch command compares 2 files based on the file size.
28	<b>CONVERT</b> This batch command converts a volume from FAT16 or FAT32 file system to NTFS file system.
29	<b>DRIVERQUERY</b> This batch command shows all installed device drivers and their properties.

30	<b>EXPAND</b> This batch command extracts files from compressed .cab cabinet files.
31	<b>FIND</b> This batch command searches for a string in files or input, outputting matching lines.
32	<b>FORMAT</b> This batch command formats a disk to use Windows-supported file system such as FAT, FAT32 or NTFS, thereby overwriting the previous content of the disk.
33	<b>HELP</b> This batch command shows the list of Windows-supplied commands.
34	<b>IPCONFIG</b> This batch command displays Windows IP Configuration. Shows configuration by connection and the name of that connection.
35	<b>LABEL</b> This batch command adds, sets or removes a disk label.
36	<b>MORE</b> This batch command displays the contents of a file or files, one screen at a time.
37	<b>NET</b> Provides various network services, depending on the command used.
38	<b>PING</b> This batch command sends ICMP/IP "echo" packets over the network to the designated address.
39	<b>SHUTDOWN</b> This batch command shuts down a computer, or logs off the current user.
40	<b>SORT</b> This batch command takes the input from a source file and sorts its contents alphabetically, from A to Z or Z to A. It prints the output on the console.
41	<b>SUBST</b> This batch command assigns a drive letter to a local folder, displays current assignments, or removes an assignment.

42	<b>SYSTEMINFO</b> This batch command shows configuration of a computer and its operating system.
43	<b>TASKKILL</b> This batch command ends one or more tasks.
44	<b>TASKLIST</b> This batch command lists tasks, including task name and process id (PID).
45	<b>XCOPY</b> This batch command copies files and directories in a more advanced way.
46	<b>TREE</b> This batch command displays a tree of all subdirectories of the current directory to any level of recursion or depth.
47	<b>FC</b> This batch command lists the actual differences between two files.
48	<b>DISKPART</b> This batch command shows and configures the properties of disk partitions.
49	<b>TITLE</b> This batch command sets the title displayed in the console window.
50	<b>SET</b> Displays the list of environment variables on the current system.



# Batch Script - VER

Advertisements

This batch command shows the version of MS-DOS you are using.

## Syntax

```
ver
```

## Example

```
@echo off  
ver
```

## Output

The output of the above command is as follows. The version number will depend upon the operating system you are working on.

```
Microsoft Windows [Version 6.3.9600]
```

# Batch Script - ASSOC

Advertisements

This is a batch command that associates an extension with a file type (FTYPE), displays existing associations, or deletes an association.

## Syntax

```
assoc - Displays all the file extensions  
assoc | find ".ext" - Displays only those file extensions which have the extension ext.
```

## Example

```
@echo off  
assoc > C:\lists.txt  
assoc | find ".doc" > C:\listsdoc.txt
```

## Output

The list of file associations will be routed to the file lists.txt. The following output shows what is there in the listsdoc.txt file after the above batch file is run.

```
.doc=Word.Document.8  
.dochtml=wordhtmlfile  
.docm=Word.DocumentMacroEnabled.12  
.docmhtml=wordmhtmlfile  
.docx=Word.Document.12  
.docxml=wordxmlfile
```

# Batch Script - CD

Advertisements

This batch command helps in making changes to a different directory, or displays the current directory.

## Syntax

```
cd
```

## Example

The following example shows how the cd command can be used in a variety of ways.

```
@echo off
Rem The cd without any parameters is used to display the current working directory
cd
Rem Changing the path to Program Files
cd\Program Files
cd
Rem Changing the path to Program Files
cd %USERPROFILE%
cd
Rem Changing to the parent directory
cd..
cd
Rem Changing to the parent directory two levels up
cd..\..
cd
```

## Output

The above command will display the following output after changing to the various folder locations.

```
C:\Users\Administrator
C:\Program Files
C:\Users\Administrator
C:\Users
C:\
```

# Batch Script - CLS

Advertisements

This batch command clears the screen.

## Syntax

```
cls
```



## Example

```
@echo off  
cls
```

## Output

The command prompt screen will be cleared.

# Batch Script - COPY

Advertisements

This batch command is used for copying files from one location to the other.

## Syntax

```
Copy [source] [destination]
```

The files will be copied from source to destination location.

## Example

The following example shows the different variants of the **copy** command.

```
@echo off
cd
Rem Copies lists.txt to the present working directory.
If there is no destination identified , it defaults to the present working directory.
copy c:\lists.txt
Rem The file lists.txt will be copied from C:\ to C:\tp location
copy C:\lists.txt c:\tp
Rem Quotation marks are required if the file name contains spaces
copy "C:\My File.txt"
Rem Copies all the files in F drive which have the txt file extension to the
current working directory copy
F:\*.txt
Rem Copies all files from dirA to dirB. Note that directories nested in dirA will not be copied
copy C:\dirA dirB
```

## Output

All actions are performed as per the remarks in the batch file.



# Batch Script - DEL

Advertisements

This batch command deletes files and not directories.

## Syntax

```
del [filename]
```

## Example

The following example shows the different variants of the **del** command.

```
@echo off
Rem Deletes the file lists.txt in C:\ 
del C:\lists.txt
Rem Deletes all files recursively in all nested directories
del /s *.txt
Rem Deletes all files recursively in all nested directories , but asks for the
confirmation from the user first
Del /p /s *.txt
```

## Output

All actions are performed as per the remarks in the batch file.



# Batch Script - DIR

Advertisements

This batch command lists the contents of a directory.

## Syntax

```
dir
```

## Example

The following example shows the different variants of the **dir** command.

```
@echo off
Rem All the directory listings from C:\ will be routed to the file lists.txt
dir C:\>C:\lists.txt
Rem Lists all directories and subdirectories recursively
dir /s
Rem Lists the contents of the directory and all subdirectories recursively, one
file per line, displaying complete path for each listed file or directory.
dir /s /b
Rem Lists all files with .txt extension.
dir *.txt
Rem Includes hidden files and system files in the listing.
dir /a
Rem Lists hidden files only.
dir /ah
```

## Output

All actions are performed as per the remarks in the batch file.



# Batch Script - DATE

Advertisements

This batch command help to find the system date.

## Syntax

```
DATE
```

## Example

```
@echo off  
echo %DATE%
```

## Output

The current date will be displayed in the command prompt. For example,

```
Mon 12/28/2015
```



# Batch Script - ECHO

Advertisements

## ECHO

This batch command displays messages, or turns command echoing on or off.

## Syntax

```
ECHO "string"
```

## Example

The following example shows the different variants of the dir command.

```
Rem Turns the echo on so that each command will be shown as executed
echo on
echo "Hello World"
```

```
Rem Turns the echo off so that each command will not be shown when executed
@echo off
echo "Hello World"
```

```
Rem Displays the contents of the PATH variable
echo %PATH%
```

## Output

The following output will be displayed in the command prompt.

```
C:\>Rem Turns the echo on so that each command will be shown as executed
```

```
C:\>echo on
```

```
C:\>echo "Hello World"
"Hello World"
```

```
C:\>Rem Turns the echo off so that each command will not be shown when executed
```

```
"Hello World"
```



# Batch Script - EXIT

Advertisements

This batch command exits the DOS console.

## Syntax

```
Exit
```

## Example

```
@echo off  
echo "Hello World"  
exit
```

## Output

The batch file will terminate and the command prompt window will close.



# Batch Script - MD

Advertisements

This batch command creates a new directory in the current location.

## Syntax

```
md [new directory name]
```

## Example

```
@echo off
md newdir
cd newdir
cd Rem "Goes back to the parent directory and create 2 directories"
cd..
md newdir1 newdir1
cd newdir1
cd
cd..
cd newdir2
cd
```

## Output

The above command produces the following output.

```
C:\newdir
C:\newdir1
C:\newdir2
```



# Batch Script - MOVE

Advertisements

This batch command moves files or directories between directories.

## Syntax

```
move [source] [destination]
```

The files will be copied from source to destination location.

## Example

The following example shows the different variants of the move command.

```
@echo off
Rem Moves the file list.txt to the directory c:\tp
move C:\lists.txt c:\tp
Rem Renames directory Dir1 to Dir2, assuming Dir1 is a directory and Dir2 does not exist.
move Dir1 Dir2
Rem Moves the file lists.txt to the current directory.
move C:\lists.txt
```

## Output

All actions are performed as per the remarks in the batch file.



# Batch Script - PATH

Advertisements

This batch command displays or sets the path variable.

## Syntax

PATH

## Example

```
@echo off  
Echo %PATH%
```

## Output

The value of the path variable will be displayed in the command prompt.



# Batch Script - PAUSE

Advertisements

This batch command prompts the user and waits for a line of input to be entered.

## Syntax

```
Pause
```

## Example

```
@echo off  
pause
```

## Output

The command prompt will show the message "Press any key to continue...." to the user and wait for the user's input.



# Batch Script - PROMPT

Advertisements

This batch command can be used to change or reset the **cmd.exe** prompt.

## Syntax

```
PROMPT [newpromptname]
```

## Example

```
@echo off  
prompt myprompt$G
```

The \$G is the greater than sign which is added at the end of the prompt.

## Output

The prompt shown to the user will now be **myprompt>**

# Batch Script - RD

Advertisements

This batch command removes directories, but the directories need to be empty before they can be removed.

## Syntax

```
rd [directoryname]
```

## Example

The following example shows the different variants of the **rd** command.

```
@echo off
Rem removes the directory called newdir
rd C:\newdir

Rem removes 2 directories
rd Dir1 Dir2

Rem Removes directory with spaces
rd "Application A"

Rem Removes the directory Dir1 including all the files and subdirectories in it rd /s Dir1
Rem Removes the directory Dir1 including all the files and subdirectories in it but
asks for a user confirmation first.
rd /q /s Dir1
```

## Output

All actions are performed as per the remarks in the batch file.



# Batch Script - REN

Advertisements

Renames files and directories

## Syntax

```
ren [oldfile dirname] [newfile dirname]
```

Renames the file name from the old file/dir name to the new one.

## Example

```
@echo off  
ren C:\lists.txt C:\newlists.txt
```

## Output

The file **lists.txt** will be renamed to **newlists.txt**.



# Batch Script - REM

Advertisements

This batch command is used for remarks in batch files, preventing the content of the remark from being executed.

## Syntax

```
REM remark description
```

## Example

```
@echo off  
REM This is a batch file
```

## Output

```
None
```



# Batch Script - START

Advertisements

This batch command starts a program in new window, or opens a document.

## Syntax

```
START "programname"
```

## Example

```
@echo off  
start notepad.exe
```

## Output

When the batch file is executed, a new notepad windows will start.



# Batch Script - TIME

Advertisements

This batch command sets or displays the time.

## Syntax

```
TIME
```

## Example

```
@echo off  
echo %TIME%
```

## Output

The current system time will be displayed. For example,

```
22:06:52.87
```



# Batch Script - TYPE

Advertisements

This batch command prints the content of a file or files to the output.

## Syntax

```
TYPE [filename]
```

Where filename is the file whose contents need to be displayed.

## Example

```
@echo off  
TYPE C:\tp\lists.txt
```

## Output

The contents of the file lists.txt will be displayed to the command prompt.



# Batch Script - VOL

Advertisements

This batch command displays the volume labels.

## Syntax

```
VOL
```

## Example

```
@echo off  
VOL
```

## Output

The output will display the current volume label. For example,

```
Volume in drive C is Windows8_OS  
Volume Serial Number is E41C-6F43
```



# Batch Script - ATTRIB

Advertisements

Displays or sets the attributes of the files in the current directory

## Syntax

```
attrib
```

## Example

The following example shows the different variants of the attrib command.

```
@echo off
Rem Displays the attributes of the file in the current directory
Attrib

Rem Displays the attributes of the file lists.txt
attrib C:\tp\lists.txt

Rem Adds the "Read-only" attribute to the file.
attrib +r C:\tp\lists.txt
Attrib C:\tp\lists.txt

Rem Removes the "Archived" attribute from the file
attrib -a C:\tp\lists.txt
Attrib C:\tp\lists.txt
```

## Output

For example,

```
A      C:\tp\assoclst.txt
A      C:\tp\List.cmd
A      C:\tp\lists.txt
A      C:\tp\listsA.txt
A      C:\tp\lists.txt
A  R    C:\tp\lists.txt
R      C:\tp\lists.txt
```



# Batch Script - CHKDSK

Advertisements

This batch command checks the disk for any problems.

## Syntax

```
chkdsk
```

## Example

```
@echo off  
chkdsk
```

## Output

The above command starts checking the current disk for any errors.

# Batch Script - CHOICE

Advertisements

This batch command provides a list of options to the user.

## Syntax

```
CHOICE /c [Options] /m [Message]
```

Where Options is the list of options to provide to the user and Message is the string message which needs to be displayed.

## Example

```
@echo off
echo "What is the file size you what"
echo "A:10MB"
echo "B:20MB"
echo "C:30MB"
choice /c ABC /m "What is your option A , B or C"
```

## Output

The above program produces the following output.

```
"What is the file size you what"
"A:10MB"
"B:20MB"
"C:30MB"
What is your option A , B or C [A,B,C]?
```



# Batch Script - CMD

Advertisements

This batch command invokes another instance of command prompt.

## Syntax

```
cmd
```

## Example

```
@echo off  
cmd
```

## Output

Another instance of command prompt will be invoked.

# Batch Script - COMP

Advertisements

This batch command compares 2 files based on the file size.

## Syntax

```
COMP [sourceA] [sourceB]
```

Wherein sourceA and sourceB are the files which need to be compared.

## Example

```
@echo off  
COMP C:\tp\lists.txt C:\tp\listsA.txt
```

## Output

The above command will compare the files lists.txt and listsA.txt and find out if the two file sizes are different.



# Batch Script - CONVERT

Advertisements

This batch command converts a volume from FAT16 or FAT32 file system to NTFS file system.

## Syntax

```
CONVERT [drive]
```

## Example

```
@echo off  
CONVERT C:\
```

## Output

The above command will convert the file system of C drive.



# Batch Script - DRIVERQUERY

Advertisements

This batch command shows all installed device drivers and their properties.

## Syntax

```
driverquery
```

## Example

```
@echo off  
driverquery
```

## Output

The above command will display the information of all the device drivers installed on the current system. Following is an example of a subset of the information displayed.

WacomPen	Wacom Serial Pen HID D Kernel	8/22/2013 4:39:15 AM
Wanarp	Remote Access IP ARP D Kernel	8/22/2013 4:35:45 AM
Wanarpv6	Remote Access IPv6 ARP Kernel	8/22/2013 4:35:45 AM
Wdf01000	Kernel Mode Driver Fra Kernel	8/22/2013 4:38:56 AM
WFPLWFS	Microsoft Windows Filt Kernel	11/9/2014 6:57:28 PM
WIMMount	WIMMount File System	8/22/2013 4:39:34 AM
WinMad	WinMad Service Kernel	5/9/2013 9:14:27 AM
WinNat	Windows NAT Driver Kernel	1/22/2014 1:10:49 AM
WinUsb	WinUsb Driver Kernel	8/22/2013 4:37:55 AM
WinVerbs	WinVerbs Service Kernel	5/9/2013 9:14:30 AM
WmiAcpi	Microsoft Windows Mana Kernel	8/22/2013 4:40:04 AM
WpdUpFltr	WPD Upper Class Filter Kernel	8/22/2013 4:38:45 AM
ws2ifsl	Windows Socket 2.0 Non Kernel	8/22/2013 4:40:03 AM
wtlmdrv	Microsoft iSCSI Target Kernel	8/22/2013 4:39:19 AM
WudfPf	User Mode Driver Frame Kernel	8/22/2013 4:37:21 AM
WUDFWpdFs	WUDFWpdFs Kernel	8/22/2013 4:36:50 AM
WUDFWpdMtp	WUDFWpdMtp Kernel	8/22/2013 4:36:50 AM



# Batch Script - EXPAND

Advertisements

This batch command extracts files from compressed .cab cabinet files.

## Syntax

```
EXPAND [cabinetfilename]
```

## Example

```
@echo off  
EXPAND excel.cab
```

## Output

The above command will extract the contents of the file excel.cab in the current location.

# Batch Script - FIND

## Advertisements

This batch command searches for a string in files or input, outputting matching lines.

## Syntax

```
FIND [text] [destination]
```

Where text is the string which needs to be searched for and destination is the source in which the search needs to take place.

## Example

```
@echo off  
FIND "Application" C:\tp\lists.txt
```

## Output

If the word “Application” resides in the file lists.txt, the line containing the string will be displayed in the command prompt.



# Batch Script - FORMAT

Advertisements

This batch command formats a disk to use Windows-supported file system such as FAT, FAT32 or NTFS, thereby overwriting the previous content of the disk.

## Syntax

```
format [drive]
```

Where drive is the drive which needs to be formatted.

## Example

```
@echo off  
format D:\
```

## Output

With the above command, D drive will be formatted.



# Batch Script - HELP

Advertisements

This batch command shows the list of Windows-supplied commands.

## Syntax

```
help
```

## Example

```
@echo off  
help
```

## Output

The above command will display a list of all commands and their description. Following is an example of a subset of the output.

SCHTASKS	Schedules commands and programs to run on a computer.
SHIFT	Shifts the position of replaceable parameters in batch files.
SHUTDOWN	Allows proper local or remote shutdown of machine.
SORT	Sorts input.
START	Starts a separate window to run a specified program or command.
SUBST	Associates a path with a drive letter.
SYSTEMINFO	Displays machine specific properties and configuration.
TASKLIST	Displays all currently running tasks including services.
TASKKILL	Kill or stop a running process or application.
TIME	Displays or sets the system time.
TITLE	Sets the window title for a CMD.EXE session.
TREE	Graphically displays the directory structure of a drive or path.
TYPE	Displays the contents of a text file.
VER	Displays the Windows version.
VERIFY	Tells Windows whether to verify that your files are written correctly to a disk.
VOL	Displays a disk volume label and serial number.
XCOPY	Copies files and directory trees.
WMIC	Displays WMI information inside interactive command shell.

# Batch Script - IPCONFIG

Advertisements

This batch command displays Windows IP Configuration. Shows configuration by connection and the name of that connection.

## Syntax

```
ipconfig
```

## Example

```
@echo off  
ipconfig
```

## Output

The above command will display the Windows IP configuration on the current machine. Following is an example of the output.

```
Windows IP Configuration  
  
Wireless LAN adapter Local Area Connection* 11:  
  Media State . . . . . : Media disconnected  
  Connection-specific DNS Suffix . :  
  
Ethernet adapter Ethernet:  
  Media State . . . . . : Media disconnected  
  Connection-specific DNS Suffix . :  
  
Wireless LAN adapter Wi-Fi:  
  Media State . . . . . : Media disconnected  
  Connection-specific DNS Suffix . :  
  
Tunnel adapter Teredo Tunneling Pseudo-Interface:  
  Media State . . . . . : Media disconnected  
  Connection-specific DNS Suffix . :
```



# Batch Script - LABEL

Advertisements

This batch command adds, sets or removes a disk label.

## Syntax

```
Label
```

## Example

```
@echo off  
label
```

## Output

The above command will prompt the user to enter a new label for the current drive.



# Batch Script - MORE

Advertisements

This batch command displays the contents of a file or files, one screen at a time.

## Syntax

```
More [filename]
```

Where filename is the file whose contents need to be listed one screen at a time.

## Example

```
@echo off  
More C:\tp\lists.txt  
Directory of C:\Program Files
```

## Output

The above command will display the contents of the file lists.txt one screen at a time. Following is an example of an output. Note the -- More (12%) – at the end of the screen. In order to proceed and display the remaining contents of the file, you need to enter a key.

```
12/22/2015 02:31 AM <DIR> .  
12/22/2015 02:31 AM <DIR> ..  
12/15/2015 11:14 PM <DIR> Application Verifier  
12/18/2015 05:06 PM <DIR> Bonjour  
12/26/2015 08:01 PM <DIR> CCleaner  
12/18/2015 05:05 PM <DIR> Common Files  
12/17/2015 11:04 AM <DIR> Git  
12/15/2015 11:09 PM <DIR> IIS  
12/15/2015 11:10 PM <DIR> IIS Express  
12/15/2015 10:16 PM <DIR> Intel  
03/18/2014 02:24 AM <DIR> Internet Explorer  
12/18/2015 05:06 PM <DIR> iPod  
12/18/2015 05:06 PM <DIR> iTunes  
12/15/2015 11:16 PM <DIR> Microsoft Identity Extensions  
12/15/2015 11:46 PM <DIR> Microsoft Office  
12/22/2015 02:31 AM <DIR> Microsoft Silverlight
```



# Batch Script - NET

Advertisements

Provides various network services, depending on the command used.

## Syntax

NET [variant]

Where its variants can be one of the following –

net accounts

net computer

net config

net continue

net file

net group

net help

net helpmsg

net localgroup

net name

net pause

net print

net send

net session

net share

net start

net statistics

net stop

net time

net use

net user

```
net view
```

## Example

```
@echo off  
Net user
```

## Output

The above command will display the current accounts defined on the system. Following is an example of an output.

```
User accounts for \\WIN-50GP30FG075
```

```
-----  
Administrator          atlbitbucket        Guest
```

```
The command completed successfully.
```



# Batch Script - PING

Advertisements

This batch command sends ICMP/IP "echo" packets over the network to the designated address.

## Syntax

```
PING [address]
```

Where address is the IP address or hostname of the destination system.

## Example

```
@echo off  
Ping 127.0.0.1
```

## Output

The above command will send ICMP/IP "echo" packets to the destination address 192.168.0.1. Following is an example of the output.

```
Pinging 127.0.0.1 with 32 bytes of data:  
Reply from 127.0.0.1: bytes = 32 time<1ms TTL = 128  
Reply from 127.0.0.1: bytes = 32 time<1ms TTL = 128  
Reply from 127.0.0.1: bytes = 32 time<1ms TTL = 128  
Reply from 127.0.0.1: bytes = 32 time<1ms TTL = 128  
  
Ping statistics for 127.0.0.1:  
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),  
Approximate round trip times in milli-seconds:  
Minimum = 0ms, Maximum = 0ms, Average = 0ms
```



# Batch Script - SHUTDOWN

Advertisements

This batch command shuts down a computer, or logs off the current user.

## Syntax

```
shutdown
```

## Example

```
@echo off  
shutdown
```

## Output

If the user executing the batch files has the relevant rights, the computer will be shutdown.



# Batch Script - SORT

Advertisements

This batch command takes the input from a source file and sorts its contents alphabetically, from A to Z or Z to A. It prints the output on the console.

## Syntax

```
Sort [filename]
```

Where filename is the file whose contents need to be sorted.

## Example

```
@echo off  
Sort C:\tp\lists.txt
```



# Batch Script - SUBST

Advertisements

This batch command assigns a drive letter to a local folder, displays current assignments, or removes an assignment.

## Syntax

```
Subst [driveletter]
```

## Example

```
@echo off  
Subst p:
```

## Output

P: will be assigned as the drive letter for the current folder.



# Batch Script - SYSTEMINFO

Advertisements

This batch command shows configuration of a computer and its operating system.

## Syntax

```
systeminfo
```

## Example

```
@echo off  
systeminfo
```

## Output

The above command will show the system information on the current system. Following is a subset of the output.

Host Name:	WIN-50GP30FG075
OS Name:	Microsoft Windows Server 2012 R2 Standard
OS Version:	6.3.9600 N/A Build 9600
OS Manufacturer:	Microsoft Corporation
OS Configuration:	Standalone Server
OS Build Type:	Multiprocessor Free
Registered Owner:	Windows User
Registered Organization:	
Product ID:	00252-70000-00000-AA535
Original Install Date:	12/13/2015, 12:10:16 AM
System Boot Time:	12/28/2015, 4:43:04 PM
System Manufacturer:	LENOVO
System Model:	20287
System Type:	x64-based PC



# Batch Script - TASKKILL

Advertisements

This batch command ends one or more tasks.

## Syntax

```
Taskkill /im [taskname]
```

## Example

```
@echo off  
Taskkill /im mspaint.exe
```

## Output

The above command will send a termination message to any open programs of MS Paint.



# Batch Script - TASKLIST

Advertisements

This batch command lists tasks, including task name and process id (PID).

## Syntax

```
Tasklist
```

## Example

```
@echo off  
Tasklist
```

## Output

The above command will list all the tasks on the current system.



# Batch Script - XCOPY

Advertisements

This batch command copies files and directories in a more advanced way.

## Syntax

```
Xcopy [source][destination]
```

## Example

```
Xcopy c:\lists.txt c:\tp\
```

## Output

The above command will copy the file lists.txt to the tp folder.



# Batch Script - TREE

Advertisements

This batch command displays a tree of all subdirectories of the current directory to any level of recursion or depth.

## Syntax

```
Tree
```

## Example

```
@echo off  
tree
```

## Output

The above command will display the tree structure of the current directory. Following is an example of the output.

```
Folder PATH listing for volume Windows8_OS  
Volume serial number is E41C-6F43  
C:  
|---newdir  
|---newdir1  
    |---newdir2
```



# Batch Script - FC

Advertisements

This batch command lists the actual differences between two files.

## Syntax

```
Fc [fileA] [fileB]
```

Where fileA and fileB are 2 files that need to be compared.

## Example

```
@echo off  
FC lists.txt listsA.txt
```

## Output

The above command will display the differences in the contents of the files (lists.txt and listsA.txt ) if any.



# Batch Script - DISKPART

Advertisements

This batch command shows and configures the properties of disk partitions.

## Syntax

```
Diskpart
```

## Example

```
@echo off  
diskpart
```

## Output

The above command shows the properties of disk partitions. Following is an example of the output.

```
Microsoft DiskPart version 6.3.9600
```

```
Copyright (C) 1999-2013 Microsoft Corporation.
```

```
On computer: WIN-50GP30FG075
```



# Batch Script - TITLE

Advertisements

This batch command sets the title displayed in the console window.

## Syntax

```
TITLE [Tilename]
```

Where tilename is the new name to be given to the title of the command prompt window.

## Example

```
@echo off  
Title "New Windows Title"
```

## Output

The above command will change the title of the window to "New Windows Title".



# Batch Script - SET

Advertisements

Displays the list of environment variables on the current system.

## Syntax

```
Set
```

## Example

```
@echo off  
set
```

## Output

The above command displays the list of environment variables on the current system.

# Batch Script - Files

In this chapter, we will learn how to create, save, execute, and modify batch files.

## Creating Batch Files

Batch files are normally created in notepad. Hence the simplest way is to open notepad and enter the commands required for the script. For this exercise, open notepad and enter the following statements.

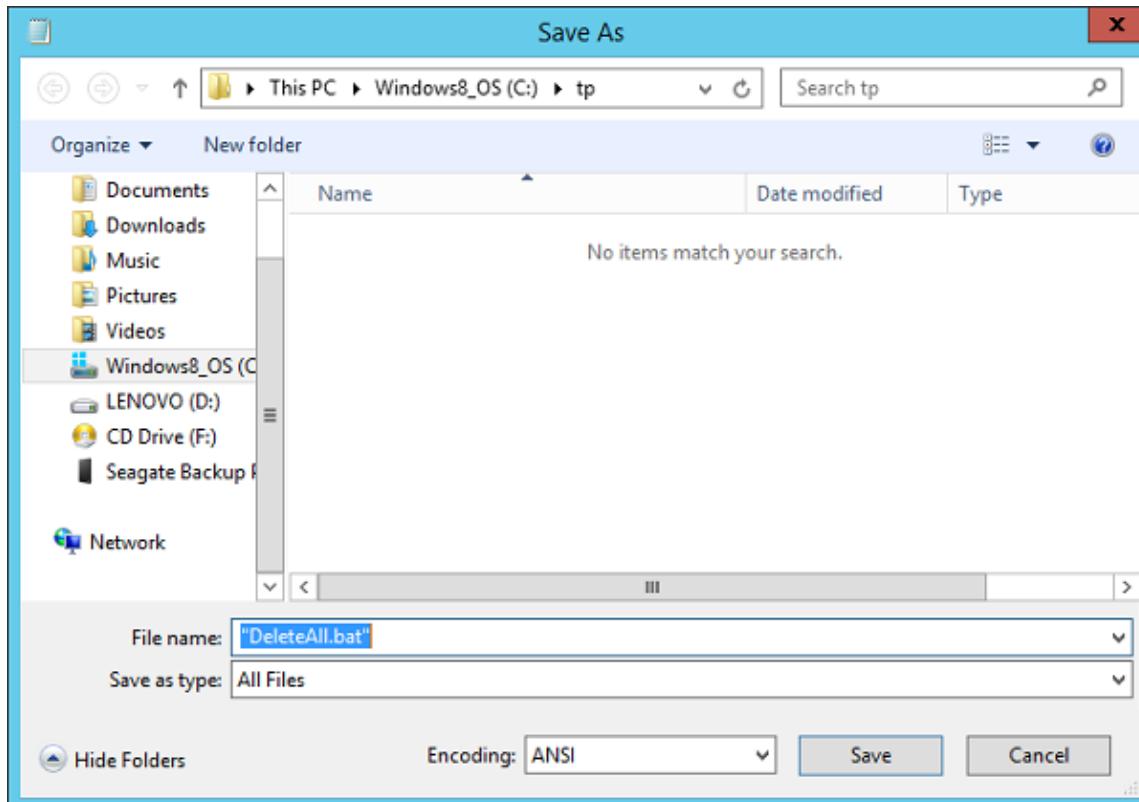
```
:: Deletes All files in the Current Directory With Prompts and Warnings  
::(Hidden, System, and Read-Only Files are Not Affected)  
::  
@ECHO OFF  
DEL .  
DR
```

# Saving Batch Files

After your batch file is created, the next step is to save your batch file. Batch files have the extension of either .bat or .cmd. Some general rules to keep in mind when naming batch files –

Try to avoid spaces when naming batch files, it sometime creates issues when they are called from other scripts.

Don't name them after common batch files which are available in the system such as ping.cmd.



The above screenshot shows how to save the batch file. When saving your batch file a few points to keep in mind.

Remember to put the .bat or .cmd at the end of the file name.

Choose the "Save as type" option as "All Files".

Put the entire file name in quotes "".

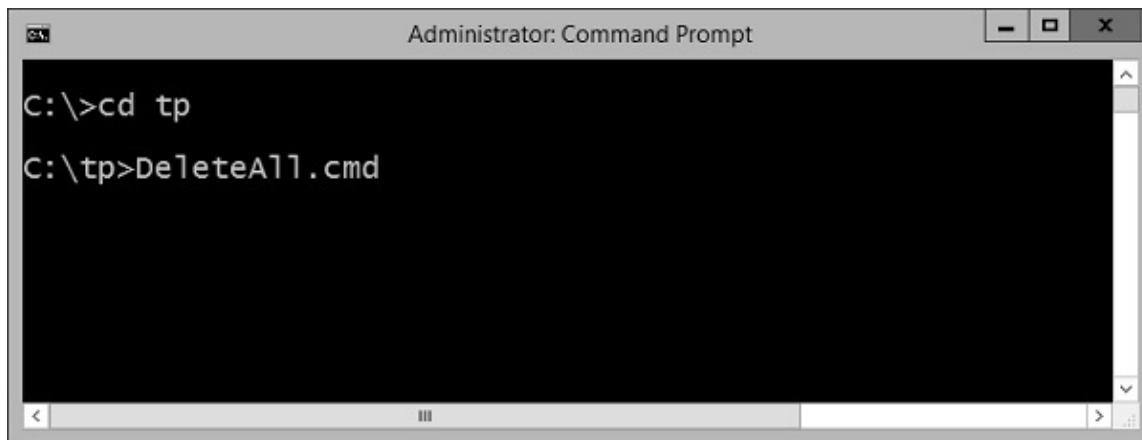
# Executing Batch Files

Following are the steps to execute a batch file –

**Step 1** – Open the command prompt (cmd.exe).

**Step 2** – Go to the location where the .bat or .cmd file is stored.

**Step 3** – Write the name of the file as shown in the following image and press the Enter button to execute the batch file.



```
C:\>cd tp
C:\tp>DeleteAll.cmd
```

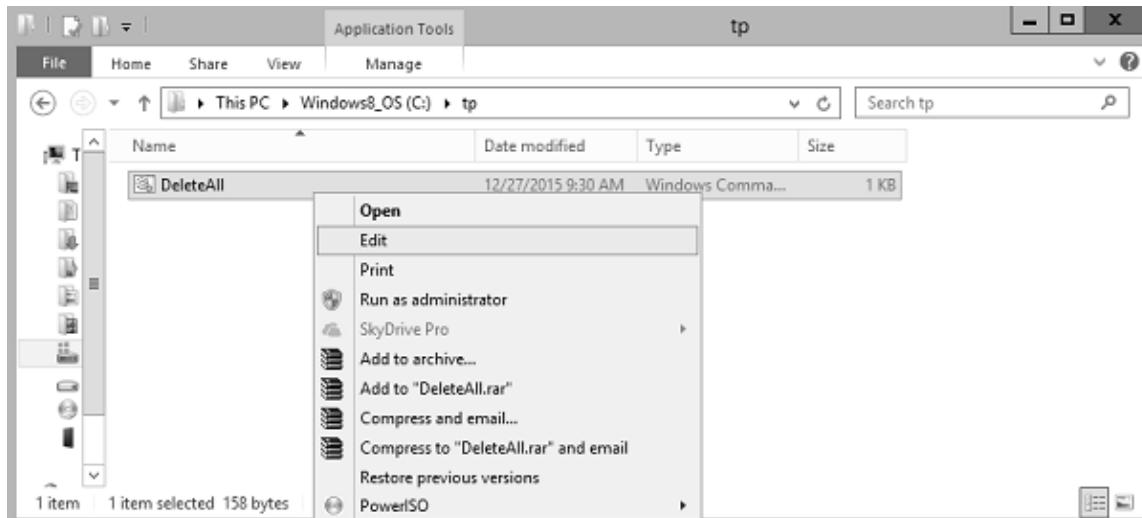
## Modifying Batch Files

Following are the steps for modifying an existing batch file.

**Step 1** – Open windows explorer.

**Step 2** – Go to the location where the .bat or .cmd file is stored.

**Step 3** – Right-click the file and choose the “Edit” option from the context menu. The file will open in Notepad for further editing.



## Batch Script - Syntax

Normally, the first line in a batch file often consists of the following command.

### ECHO Command

```
@echo off
```

By default, a batch file will display its command as it runs. The purpose of this first command is to turn off this display. The command "echo off" turns off the display for the

whole script, except for the "echo off" command itself. The "at" sign "@" in front makes the command apply to itself as well.

## Documentation

Very often batch files also contains lines that start with the "Rem" command. This is a way to enter comments and documentation. The computer ignores anything on a line following Rem. For batch files with increasing amount of complexity, this is often a good idea to have comments.

## First Batch Script Program

Let's construct our simple first batch script program. Open notepad and enter the following lines of code. Save the file as "List.cmd".

The code does the following –

Uses the echo off command to ensure that the commands are not shown when the code is executed.

The Rem command is used to add a comment to say what exactly this batch file does.

The dir command is used to take the contents of the location C:\Program Files.

The '>' command is used to redirect the output to the file C:\lists.txt.

Finally, the echo command is used to tell the user that the operation is completed.

```
@echo off
Rem This is for listing down all the files in the directory Program files
dir "C:\Program Files" > C:\lists.txt
echo "The program has completed"
```

When the above command is executed, the names of the files in C:\Program Files will be sent to the file C:\Lists.txt and in the command prompt the message "The program has completed" will be displayed.

## Batch Script - Variables

There are two types of variables in batch files. One is for parameters which can be passed when the batch file is called and the other is done via the set command.

## Command Line Arguments

Batch scripts support the concept of command line arguments wherein arguments can be passed to the batch file when invoked. The arguments can be called from the batch files through the variables %1, %2, %3, and so on.

The following example shows a batch file which accepts 3 command line arguments and

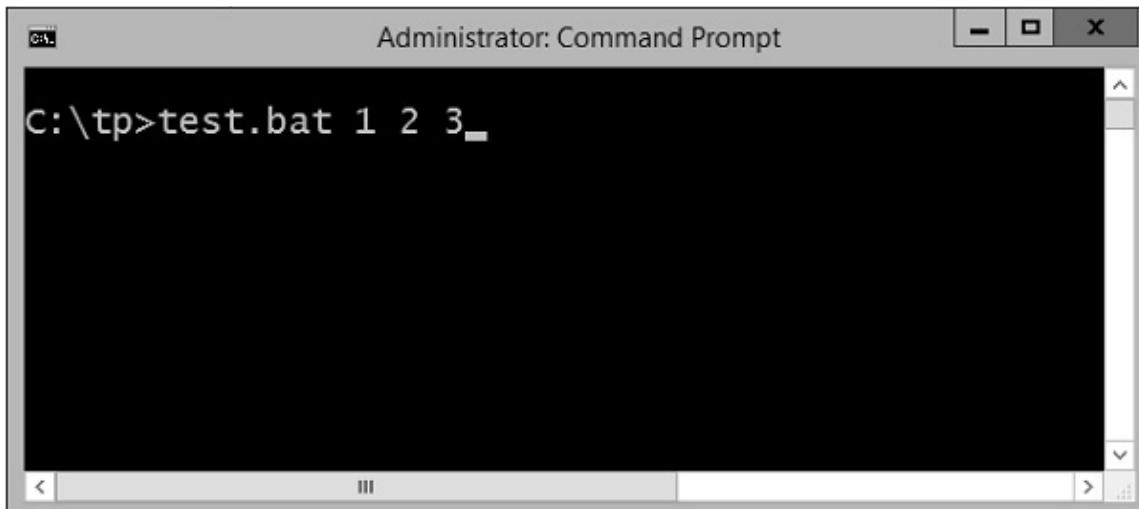
echo's them to the command line screen.

```
@echo off  
echo %1  
echo %2  
echo %3
```

If the above batch script is stored in a file called test.bat and we were to run the batch as

```
Test.bat 1 2 3
```

Following is a screenshot of how this would look in the command prompt when the batch file is executed.



The above command produces the following output.

```
1  
2  
3
```

If we were to run the batch as

```
Example 1 2 3 4
```

The output would still remain the same as above. However, the fourth parameter would be ignored.

## Set Command

The other way in which variables can be initialized is via the 'set' command. Following is the syntax of the set command.

### Syntax

```
set /A variable-name=value
```

where,

**variable-name** is the name of the variable you want to set.

**value** is the value which needs to be set against the variable.

**/A** – This switch is used if the value needs to be numeric in nature.

The following example shows a simple way the set command can be used.

## Example

```
@echo off  
set message=Hello World  
echo %message%
```

In the above code snippet, a variable called message is defined and set with the value of "Hello World".

To display the value of the variable, note that the variable needs to be enclosed in the % sign.

## Output

The above command produces the following output.

```
Hello World
```

## Working with Numeric Values

In batch script, it is also possible to define a variable to hold a numeric value. This can be done by using the /A switch.

The following code shows a simple way in which numeric values can be set with the /A switch.

```
@echo off  
SET /A a = 5  
SET /A b = 10  
SET /A c = %a% + %b%  
echo %c%
```

We are first setting the value of 2 variables, a and b to 5 and 10 respectively.

We are adding those values and storing in the variable c.

Finally, we are displaying the value of the variable c.

The output of the above program would be 15.

All of the arithmetic operators work in batch files. The following example shows arithmetic operators can be used in batch files.

```
@echo off  
SET /A a = 5  
SET /A b = 10  
SET /A c = %a% + %b%  
echo %c%
```

```
SET /A c = %a% - %b%
echo %c%
SET /A c = %b% / %a%
echo %c%
SET /A c = %b% * %a%
echo %c%
```

The above command produces the following output.

```
15
-5
2
50
```

## Local vs Global Variables

In any programming language, there is an option to mark variables as having some sort of scope, i.e. the section of code on which they can be accessed. Normally, variable having a global scope can be accessed anywhere from a program whereas local scoped variables have a defined boundary in which they can be accessed.

DOS scripting also has a definition for locally and globally scoped variables. By default, variables are global to your entire command prompt session. Call the SETLOCAL command to make variables local to the scope of your script. After calling SETLOCAL, any variable assignments revert upon calling ENDLOCAL, calling EXIT, or when execution reaches the end of file (EOF) in your script. The following example shows the difference when local and global variables are set in the script.

### Example

```
@echo off
set globalvar = 5
SETLOCAL
set var = 13145
set /A var = %var% + 5
echo %var%
echo %globalvar%
ENDLOCAL
```

Few key things to note about the above program.

The 'globalvar' is defined with a global scope and is available throughout the entire script.

The 'var' variable is defined in a local scope because it is enclosed between a 'SETLOCAL' and 'ENDLOCAL' block. Hence, this variable will be destroyed as soon the 'ENDLOCAL' statement is executed.

### Output

The above command produces the following output.

13150

5

You will notice that the command echo %var% will not yield anything because after the ENDLOCAL statement, the 'var' variable will no longer exist.

## Working with Environment Variables

If you have variables that would be used across batch files, then it is always preferable to use environment variables. Once the environment variable is defined, it can be accessed via the % sign. The following example shows how to see the JAVA\_HOME defined on a system. The JAVA\_HOME variable is a key component that is normally used by a wide variety of applications.

```
@echo off  
echo %JAVA_HOME%
```

The output would show the JAVA\_HOME directory which would depend from system to system. Following is an example of an output.

```
C:\Atlassian\Bitbucket\4.0.1\jre
```

## Batch Script - Comments

It's always a good practice to add comments or documentation for the scripts which are created. This is required for maintenance of the scripts to understand what the script actually does.

For example, consider the following piece of code which has no form of comments. If any average person who has not developed the following script tries to understand the script, it would take a lot of time for that person to understand what the script actually does.

```
ECHO OFF  
IF NOT "%OS%"=="Windows_NT" GOTO Syntax  
ECHO.%* | FIND "?" >NUL  
IF NOT ERRORLEVEL 1 GOTO Syntax  
IF NOT [%2]==[] GOTO Syntax  
SETLOCAL  
SET WSS=  
IF NOT [%1]==[] FOR /F "tokens = 1 delims = \ " %%A IN ('ECHO.%~1') DO SET WSS = %%A  
FOR /F "tokens = 1 delims = \ " %%a IN ('NET VIEW ^| FIND /I "\%WSS%"') DO FOR /F  
"tokens = 1 delims = " %%a IN ('NBTSTAT -a %%a ^| FIND /I /V "%a" ^| FIND "<03>"')  
DO ECHO.%%a %%A  
ENDLOCAL  
GOTO:EOF  
ECHO Display logged on users and their workstations.  
ECHO Usage: ACTUSR [ filter ]  
IF "%OS%"=="Windows_NT" ECHO Where: filter is the first part  
of the computer name^(s^) to be displayed
```

## Comments Using the Rem Statement

There are two ways to create comments in Batch Script; one is via the Rem command. Any text which follows the Rem statement will be treated as comments and will not be executed. Following is the general syntax of this statement.

## Syntax

```
Rem Remarks
```

where 'Remarks' is the comments which needs to be added.

The following example shows a simple way the **Rem** command can be used.

## Example

```
@echo off
Rem This program just displays Hello World
set message=Hello World
echo %message%
```

## Output

The above command produces the following output. You will notice that the line with the Rem statement will not be executed.

```
Hello World
```

## Comments Using the :: Statement

The other way to create comments in Batch Script is via the :: command. Any text which follows the :: statement will be treated as comments and will not be executed. Following is the general syntax of this statement.

## Syntax

```
:: Remarks
```

where 'Remarks' is the comment which needs to be added.

The following example shows a simple way the Rem command can be used.

## Example

```
@echo off
:: This program just displays Hello World
set message = Hello World
echo %message%
```

## Output

The above command produces the following output. You will notice that the line with the :: statement will not be executed.

```
Hello World
```

**Note** – If you have too many lines of Rem, it could slow down the code, because in the end each line of code in the batch file still needs to be executed.

Let's look at the example of the large script we saw at the beginning of this topic and see how it looks when documentation is added to it.

```
=====
:: The below example is used to find computer and logged on users
::
=====
ECHO OFF
:: Windows version check
IF NOT "%OS%"=="Windows_NT" GOTO Syntax
ECHO.%* | FIND "?" >NUL
:: Command line parameter check
IF NOT ERRORLEVEL 1 GOTO Syntax
IF NOT [%2]==[] GOTO Syntax
:: Keep variable local
SETLOCAL
:: Initialize variable
SET WSS=
:: Parse command line parameter
IF NOT [%1]==[] FOR /F "tokens = 1 delims = \ " %%A IN ('ECHO.%~1') DO SET WSS = %%A
:: Use NET VIEW and NBTSTAT to find computers and logged on users
FOR /F "tokens = 1 delims = \ " %%a IN ('NET VIEW ^| FIND /I "\%WSS%") DO FOR /F
"tokens = 1 delims = " %%A IN ('NBTSTAT -a %%a ^| FIND /I /V "%a" ^| FIND
"<03>") DO ECHO.%a %%A
:: Done
ENDLOCAL
GOTO:EOF
:Syntax
ECHO Display logged on users and their workstations.
ECHO Usage: ACTUSR [ filter ]
IF "%OS%"=="Windows_NT" ECHO Where: filter is the first part of the
computer name^(s^) to be displayed
```

You can now see that the code has become more understandable to users who have not developed the code and hence is more maintainable.

## Batch Script - Strings

In DOS, a string is an ordered collection of characters, such as "Hello, World!".

S.No	Strings & Description
1	<b>Create String</b> A string can be created in DOS in the following way.
2	<b>Empty String</b> Empty String
3	<b>String Interpolation</b> String interpolation is a way to construct a new String value from a mix of constants, variables, literals, and expressions by including their values inside a

string literal.

#### **String Concatenation**

- 4 You can use the set operator to concatenate two strings or a string and a character, or two characters. Following is a simple example which shows how to use string concatenation.

#### **String length**

- 5 In DOS scripting, there is no length function defined for finding the length of a string. There are custom-defined functions which can be used for the same. Following is an example of a custom-defined function for seeing the length of a string.

#### **toInt**

- 6 A variable which has been set as string using the set variable can be converted to an integer using the /A switch which is using the set variable. The following example shows how this can be accomplished.

#### **Align Right**

- 7 This used to align text to the right, which is normally used to improve readability of number columns.

#### **Left String**

- 8 This is used to extract characters from the beginning of a string.

#### **Mid String**

- 9 This is used to extract a substring via the position of the characters in the string.

#### **Remove**

- 10 The string substitution feature can also be used to remove a substring from another string.

#### **Remove Both Ends**

- 11 This is used to remove the first and the last character of a string.

#### **Remove All Spaces**

- 12 This is used to remove all spaces in a string via substitution.

#### **Replace a String**

- 13 To replace a substring with another string use the string substitution feature.

**Right String**

- 14 This is used to extract characters from the end of a string.



# Batch Script - Create String

Advertisements

A string can be created in DOS in the following way.

## Example

```
@echo off  
:: This program just displays Hello World  
set message = Hello World  
echo %message%
```

## Output

The above command produces the following output.

```
Hello World
```

# Batch Script - Empty String

Advertisements

An empty string can be created in DOS Scripting by assigning it no value during its initialization as shown in the following example.

```
Set a=
```

To check for an existence of an empty string, you need to encompass the variable name in square brackets and also compare it against a value in square brackets as shown in the following example.

```
[%a%]==[]
```

The following example shows how an empty string can be created and how to check for the existence of an empty string.

## Example

```
@echo off
SET a=
SET b = Hello
if [%a%]==[] echo "String A is empty"
if [%b%]==[] echo "String B is empty "
```

## Output

The above command produces the following output.

```
String A is empty
```

# Batch Script - String Interpolation

Advertisements

String interpolation is a way to construct a new String value from a mix of constants, variables, literals, and expressions by including their values inside a string literal.

In DOS scripting, the string interpolation can be done using the set command and lining up the numeric defined variables or any other literals in one line when using the set command.

The following example shows how a string interpolation can be done with numeric values as well.

## Example

```
@echo off
SET a = Hello
SET b = World
SET /A d = 50
SET c=%a% and %b% %d%
echo %c%
```

## Output

The above command produces the following output.

```
Hello and World 50
```

# Batch Script - String Concatenation

Advertisements

You can use the set operator to concatenate two strings or a string and a character, or two characters. Following is a simple example which shows how to use string concatenation.

## Example

```
@echo off  
SET a = Hello  
SET b = World  
SET c=%a% and %b%  
echo %c%
```

## Output

The above command produces the following output.

```
Hello and World
```

# Batch Script - String length

Advertisements

In DOS scripting, there is no length function defined for finding the length of a string. There are custom-defined functions which can be used for the same. Following is an example of a custom-defined function for seeing the length of a string.

## Example

```
@echo off
set str = Hello World
call :strLen str strlen
echo String is %strlen% characters long
exit /b

:strLen
setlocal enabledelayedexpansion

:strLen_Loop
if not "%!1:~%len%!"==""
  set /A len+=1 & goto :strLen_Loop
(endlocal & set %2=%len%)
goto :eof
```

A few key things to keep in mind about the above program are –

The actual code which finds the length of string is defined in the :strLen block.

The length of the string is maintained in the variable len.

## Output

The above command produces the following output.

# Batch Script - toInt

Advertisements

A variable which has been set as string using the set variable can be converted to an integer using the /A switch which is using the set variable. The following example shows how this can be accomplished.

## Example

```
@echo off
set var = 13145
set /A var=%var% + 5
echo %var%
```

## Output

The above command produces the following output.

```
13150
```

Apart from this, strings have the following implementations which are available. Batch scripts have the following commands which are used to carry out string manipulation in strings.

```
%variable:~num_chars_to_skip%
%variable:~num_chars_to_skip,num_chars_to_keep%
```

This can include negative numbers –

```
%variable:~num_chars_to_skip, -num_chars_to_keep%
%variable:~-num_chars_to_skip,num_chars_to_keep%
%variable:~-num_chars_to_skip,-num_chars_to_keep%
```

Let us discuss the possible string operations that can be performed by using the above commands.

# Batch Script - Align Right

Advertisements

This used to align text to the right, which is normally used to improve readability of number columns.

## Example

```
@echo off
set x = 1000
set y = 1
set y = %y%
echo %x%

set y = %y:~-4%
echo %y%
```

A few key things to note about the above program is –

Spaces are added to the variable of y, in this case we are adding 9 spaces to the variable of y.

We are using the ~-4 option to say that we just want to show the last 4 characters of the string y.

## Output

The above command produces the following output. The key thing to note is that the value of 2 is aligned to match the units columns when displaying numbers.

```
1000
1
```

# Batch Script - Left String

Advertisements

This is used to extract characters from the beginning of a string.

## Example

```
@echo off  
set str = Helloworld  
echo %str%  
  
set str = %str:~0,5%  
echo %str%
```

The key thing to note about the above program is, `~0,5` is used to specify the characters which needs to be displayed. In this case, we are saying character 0 to 5 should be displayed.

## Output

The above command produces the following output.

```
Helloworld  
Hello
```



# Batch Script - Mid String

Advertisements

This is used to extract a substring via the position of the characters in the string.

## Example

```
@echo off  
set str = Helloworld  
echo %str%  
  
set str = %str:~5,10%  
echo %str%
```

The key thing to note about the above program is, `~5,10` is used to specify the characters which needs to be displayed. In this case, we want character 5 to 10 should be displayed.

## Output

The above command produces the following output.

```
Helloworld  
world
```



# Batch Script - Remove

Advertisements

The string substitution feature can also be used to remove a substring from another string.

## Example

```
@echo off  
set str = Batch scripts is easy. It is really easy.  
echo %str%  
  
set str = %str:is = %  
echo %str%
```

The key thing to note about the above program is, the 'is' word is being removed from the string using the ':stringtoberemoved' = command.

## Output

The above command produces the following output.

```
Batch scripts is easy. It is really easy.  
Batch scripts easy. It really easy.
```



# Batch Script - Remove Both Ends

Advertisements

This is used to remove the first and the last character of a string

## Example

```
@echo off
set str = Batch scripts is easy. It is really easy
echo %str%

set str = %str:~1,-1%
echo %str%
```

The key thing to note about the above program is, the ~1,-1 is used to remove the first and last character of a string.

## Output

The above command produces the following output.

```
Batch scripts is easy. It is really easy
atch scripts is easy. It is really eas
```



# Batch Script - Remove All Spaces

Advertisements

This is used to remove all spaces in a string via substitution.

## Example

```
@echo off  
set str = This string      has      a    lot   of spaces  
echo %str%  
  
set str=%str:=%  
echo %str%
```

The key thing to note about the above program is, the : = operator is used to remove all spaces from a string.

## Output

The above command produces the following output.

```
This string      has      a    lot   of spaces  
Thisstringhasalotofspaces
```



# Batch Script - Replace a String

Advertisements

To replace a substring with another string use the string substitution feature.

## Example

```
@echo off  
set str = This message needs changed.  
echo %str%  
  
set str = %str:needs = has%  
echo %str%
```

The key thing to note about the above program is, the example replaces the word 'needs' with the string 'has' via the statement %str:needs = has%

## Output

The above command produces the following output.

```
This message needs changed.  
This message has changed.
```



# Batch Script - Right String

Advertisements

This is used to extract characters from the end of a string.

## Example

```
@echo off  
set str = This message needs changed.  
echo %str%  
  
set str = %str:~-8%  
echo %str%
```

The key thing to note about the above program is, the right hand of the string is extracted by using the ~-'number of characters to extract' operator.

## Output

The above command produces the following output.

```
This message needs changed.  
changed.
```

# Batch Script - Arrays

Arrays are not specifically defined as a type in Batch Script but can be implemented. The following things need to be noted when arrays are implemented in Batch Script.

Each element of the array needs to be defined with the set command.

The 'for' loop would be required to iterate through the values of the array.

## Creating an Array

An array is created by using the following set command.

```
set a[0]=1
```

Where 0 is the index of the array and 1 is the value assigned to the first element of the array.

Another way to implement arrays is to define a list of values and iterate through the list of values. The following example show how this can be implemented.

### Example

```
@echo off
set list = 1 2 3 4
(for %%a in (%list%) do (
    echo %%a
))
```

### Output

The above command produces the following output.

```
1
2
3
4
```

## Accessing Arrays

You can retrieve a value from the array by using subscript syntax, passing the index of the value you want to retrieve within square brackets immediately after the name of the array.

### Example

```
@echo off
set a[0]=1
```

```
echo %a[0]%
```

In this example, the index starts from 0 which means the first element can be accessed using index as 0, the second element can be accessed using index as 1 and so on. Let's check the following example to create, initialize and access arrays –

```
@echo off
set a[0] = 1
set a[1] = 2
set a[2] = 3
echo The first element of the array is %a[0]%
echo The second element of the array is %a[1]%
echo The third element of the array is %a[2]%
```

The above command produces the following output.

```
The first element of the array is 1
The second element of the array is 2
The third element of the array is 3
```

## Modifying an Array

To add an element to the end of the array, you can use the set element along with the last index of the array element.

### Example

```
@echo off
set a[0] = 1
set a[1] = 2
set a[2] = 3
Rem Adding an element at the end of an array
Set a[3] = 4
echo The last element of the array is %a[3]%
```

The above command produces the following output.

```
The last element of the array is 4
```

You can modify an existing element of an Array by assigning a new value at a given index as shown in the following example –

```
@echo off
set a[0] = 1
set a[1] = 2
set a[2] = 3
Rem Setting the new value for the second element of the array
Set a[1] = 5
echo The new value of the second element of the array is %a[1]%
```

The above command produces the following output.

```
The new value of the second element of the array is 5
```

# Iterating Over an Array

Iterating over an array is achieved by using the 'for' loop and going through each element of the array. The following example shows a simple way that an array can be implemented.

```
@echo off
setlocal enabledelayedexpansion
set topic[0] = comments
set topic[1] = variables
set topic[2] = Arrays
set topic[3] = Decision making
set topic[4] = Time and date
set topic[5] = Operators

for /l %%n in (0,1,5) do (
    echo !topic[%n]!
)
```

Following things need to be noted about the above program –

Each element of the array needs to be specifically defined using the set command.

The 'for' loop with the /L parameter for moving through ranges is used to iterate through the array.

## Output

The above command produces the following output.

```
Comments
variables
Arrays
Decision making
Time and date
Operators
```

# Length of an Array

The length of an array is done by iterating over the list of values in the array since there is no direct function to determine the number of elements in an array.

```
@echo off
set Arr[0] = 1
set Arr[1] = 2
set Arr[2] = 3
set Arr[3] = 4
set "x = 0"
:SymLoop

if defined Arr[%x%] (
    call echo %%Arr[%x%]%%
    set /a "x+=1"
    GOTO :SymLoop
)
```

```
echo "The length of the array is" %x%
```

## Output

The above command produces the following output.

```
The length of the array is 4
```

## Creating Structures in Arrays

Structures can also be implemented in batch files using a little bit of an extra coding for implementation. The following example shows how this can be achieved.

### Example

```
@echo off
set len = 3
set obj[0].Name = Joe
set obj[0].ID = 1
set obj[1].Name = Mark
set obj[1].ID = 2
set obj[2].Name = Mohan
set obj[2].ID = 3
set i = 0
:loop

if %i% equ %len% goto :eof
set cur.Name=
set cur.ID=

for /f "usebackq delims==.tokens=1-3" %%j in (`set obj[%i%]`) do (
    set cur.%~k=%~l
)
echo Name = %cur.Name%
echo Value = %cur.ID%
set /a i = %i%+1
goto loop
```

The following key things need to be noted about the above code.

Each variable defined using the `set` command has 2 values associated with each index of the array.

The variable `i` is set to 0 so that we can loop through the structure will the length of the array which is 3.

We always check for the condition on whether the value of `i` is equal to the value of `len` and if not, we loop through the code.

We are able to access each element of the structure using the `obj[%i%]` notation.

## Output

The above command produces the following output.

```
Name = Joe
```

```
Value = 1
Name = Mark
Value = 2
Name = Mohan
Value = 3
```

## Batch Script - Decision Making

Decision-making structures require that the programmer specify one or more conditions to be evaluated or tested by the program, along with a statement or statements to be executed if the condition is determined to be **true**, and optionally, other statements to be executed if the condition is determined to be **false**.

S.No	Strings & Description
1	<b>If Statement</b> The first decision-making statement is the 'if' statement.
2	<b>If/else Statement</b> The next decision making statement is the If/else statement. Following is the general form of this statement.
3	<b>Nested If Statements</b> Sometimes, there is a requirement to have multiple 'if' statement embedded inside each other. Following is the general form of this statement.

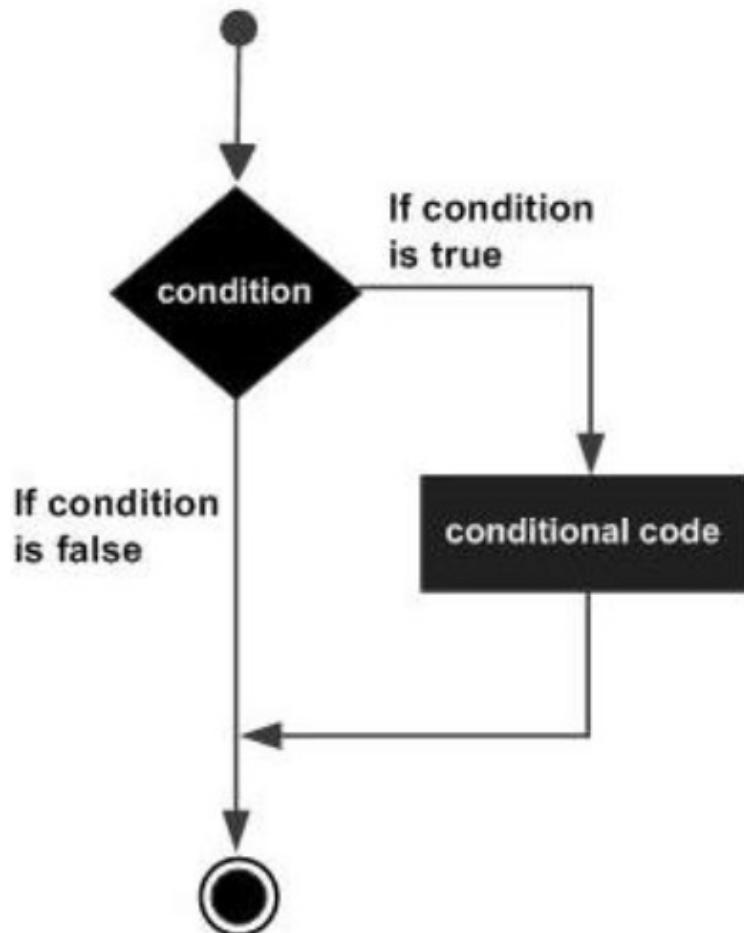
# Batch Script - If Statement

Advertisements

The first decision-making statement is the 'if' statement. The general form of this statement in Batch Script is as follows –

```
if(condition) do_something
```

The general working of this statement is that first a condition is evaluated in the 'if' statement. If the condition is true, it then executes the statements. The following diagram shows the flow of the **if** statement.



## Checking Variables

One of the common uses for the 'if' statement in Batch Script is for checking variables which

are set in Batch Script itself. The evaluation of the 'if' statement can be done for both strings and numbers.

## Checking Integer Variables

The following example shows how the 'if' statement can be used for numbers.

### Example

```
@echo off
SET /A a = 5
SET /A b = 10
SET /A c = %a% + %b%
if %c%==15 echo "The value of variable c is 15"
if %c%==10 echo "The value of variable c is 10"
```

The key thing to note about the above program is –

The first 'if' statement checks if the value of the variable c is 15. If so, then it echo's a string to the command prompt.

Since the condition in the statement - if %c% == 10 echo "The value of variable c is 10 evaluates to false, the echo part of the statement will not be executed.

### Output

The above command produces the following output.

```
15
```

## Checking String Variables

The following example shows how the 'if' statement can be used for strings.

### Example

```
@echo off
SET str1 = String1
SET str2 = String2
if %str1%==String1 echo "The value of variable String1"
if %str2%==String3 echo "The value of variable c is String3"
```

The key thing to note about the above program is –

The first 'if' statement checks if the value of the variable str1 contains the string "String1". If so, then it echo's a string to the command prompt.

Since the condition of the second 'if' statement evaluates to false, the echo part of the statement will not be executed.

### Output

The above command produces the following output.

```
"The value of variable String1"
```

**Note** – One key thing to note is that the evaluation in the 'if' statement is "case-sensitive". The same program as above is modified a little as shown in the following example. In the first statement, we have changed the comparison criteria. Because of the different casing, the output of the following program would yield nothing.

```
@echo off
SET str1 = String1
SET str2 = String2
if %str1%==StrinG1 echo "The value of variable String1"
if %str2%==String3 echo "The value of variable c is String3"
```

## Checking Command Line Arguments

Another common use of the 'if' statement is used to check for the values of the command line arguments which are passed to the batch files. The following example shows how the 'if' statement can be used to check for the values of the command line arguments.

### Example

```
@echo off
echo %1
echo %2
echo %3
if %1%==1 echo "The value is 1"
if %2%==2 echo "The value is 2"
if %3%==3 echo "The value is 3"
```

The key thing to note about the above program is –

The above program assumes that 3 command line arguments will be passed when the batch script is executed.

A comparison is done for each command line argument against a value. If the criteria passes then a string is sent as the output.

### Output

If the above code is saved in a file called test.bat and the program is executed as

```
test.bat 1 2 3
```

Following will be the output of the above program.

```
1
2
3
"The value is 1"
"The value is 2"
"The value is 3"
```

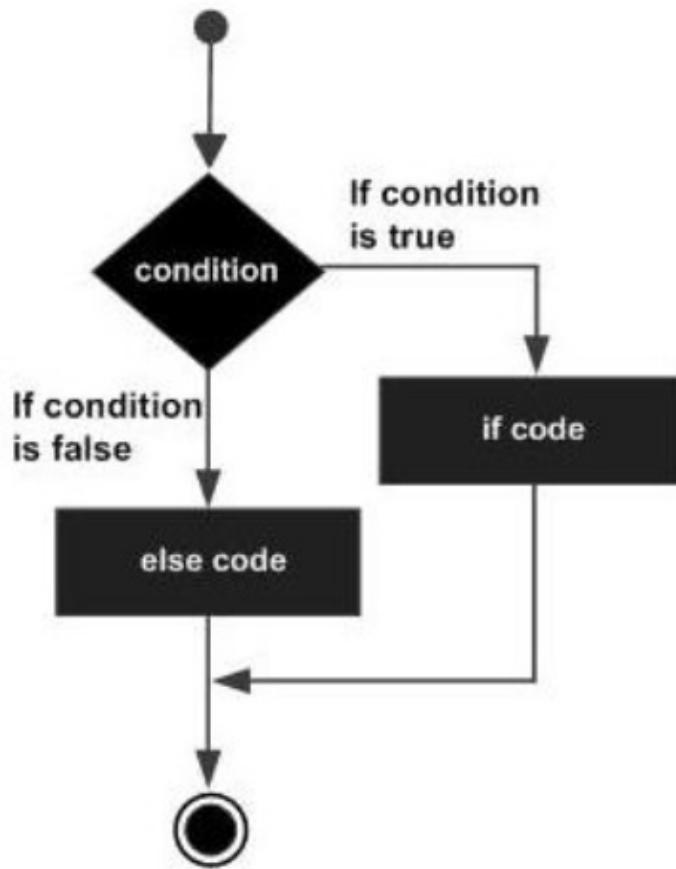
# Batch Script - If/else Statement

Advertisements

The next decision making statement is the If/else statement. Following is the general form of this statement.

```
If (condition) (do_something) ELSE (do_something_else)
```

The general working of this statement is that first a condition is evaluated in the 'if' statement. If the condition is true, it then executes the statements thereafter and stops before the else condition and exits out of the loop. If the condition is false, it then executes the statements in the else statement block and then exits the loop. The following diagram shows the flow of the 'if' statement.



## Checking Variables

Just like the 'if' statement in Batch Script, the if-else can also be used for checking variables

which are set in Batch Script itself. The evaluation of the 'if' statement can be done for both strings and numbers.

## Checking Integer Variables

The following example shows how the 'if' statement can be used for numbers.

### Example

```
@echo off
SET /A a = 5
SET /A b = 10
SET /A c = %a% + %b%
if %c%==15 (echo "The value of variable c is 15") else (echo "Unknown value")
if %c%==10 (echo "The value of variable c is 10") else (echo "Unknown value")
```

The key thing to note about the above program is –

Each 'if else' code is placed in the brackets (). If the brackets are not placed to separate the code for the 'if and else' code, then the statements would not be valid proper if else statements.

In the first 'if else' statement, the if condition would evaluate to true.

In the second 'if else' statement, the else condition will be executed since the criteria would be evaluated to false.

### Output

The above command produces the following output.

```
"The value of variable c is 15"
"Unknown value"
```

## Checking String Variables

The same example can be repeated for strings. The following example shows how the 'if else' statement can be used to strings.

### Example

```
@echo off
SET str1 = String1
SET str2 = String2

if %str1%==String1 (echo "The value of variable String1") else (echo "Unknown value")
if %str2%==String3 (echo "The value of variable c is String3") else (echo "Unknown value")
```

The key thing to note about the above program is –

The first 'if' statement checks if the value of the variable str1 contains the string "String1". If so, then it echo's a string to the command prompt.

Since the condition of the second 'if' statement evaluates to false, the echo part of

the statement will not be executed.

## Output

The above command produces the following output.

```
"The value of variable String1"  
"Unknown value"
```

## Checking Command Line Arguments

The 'if else' statement can also be used for checking of command line arguments. The following example show how the 'if' statement can be used to check for the values of the command line arguments.

### Example

```
@echo off  
echo %1  
echo %2  
echo %3  
if %1%==1 (echo "The value is 1") else (echo "Unknown value")  
if %2%==2 (echo "The value is 2") else (echo "Unknown value")  
if %3%==3 (echo "The value is 3") else (echo "Unknown value")
```

## Output

If the above code is saved in a file called test.bat and the program is executed as

```
test.bat 1 2 4
```

Following will be the output of the above program.

```
1  
2  
4  
"The value is 1"  
"The value is 2"  
"Unknown value"
```

## if defined

A special case for the 'if' statement is the "if defined", which is used to test for the existence of a variable. Following is the general syntax of the statement.

```
if defined somevariable somecommand
```

Following is an example of how the 'if defined' statement can be used.

### Example

```
@echo off  
SET str1 = String1
```

```
SET str2 = String2
if defined str1 echo "Variable str1 is defined"

if defined str3 (echo "Variable str3 is defined") else (echo "Variable str3 is not defined")
```

## Output

The above command produces the following output.

```
"Variable str1 is defined"
"Variable str3 is not defined"
```

## if exists

Another special case for the 'if' statement is the "if exists ", which is used to test for the existence of a file. Following is the general syntax of the statement.

```
If exist somefile.ext do_something
```

Following is an example of how the 'if exists' statement can be used.

## Example

```
@echo off
if exist C:\set2.txt echo "File exists"
if exist C:\set3.txt (echo "File exists") else (echo "File does not exist")
```

## Output

Let's assume that there is a file called set2.txt in the C drive and that there is no file called set3.txt. Then, following will be the output of the above code.

```
"File exists"
"File does not exist"
```



# Batch Script - Nested If Statements

Advertisements

Sometimes, there is a requirement to have multiple 'if' statement embedded inside each other. Following is the general form of this statement.

```
if(condition1) if (condition2) do_something
```

So only if condition1 and condition2 are met, will the code in the do\_something block be executed.

Following is an example of how the nested if statements can be used.

## Example

```
@echo off
SET /A a = 5
SET /A b = 10
if %a%==5 if %b%==10 echo "The value of the variables are correct"
```

## Output

The above command produces the following output.

```
"The value of the variables are correct"
```

## If errorlevel

Yet another special case is "if errorlevel", which is used to test the exit codes of the last command that was run. Various commands issue integer exit codes to denote the status of the command. Generally, commands pass 0 if the command was completed successfully and 1 if the command failed.

Following is the general syntax of this statement.

```
if errorlevel n somecommand
```

where "n" is one of the integer exit codes.

## Goto Statement

Generally, the execution of a batch file proceeds line-by-line with the command(s) on each

line being run in turn. However, it is often desirable to execute a particular section of a batch file while skipping over other parts. The capability to hop to a particular section is provided by the appropriately named "goto" command (written as one word). The target section is labeled with a line at the beginning that has a name with a leading colon. Thus the script looks like –

```
...
goto :label
...some commands
:label
...some other commands
```

Execution will skip over "some commands" and start with "some other commands". The label can be a line anywhere in the script, including before the "goto" command. "Goto" commands often occur in "if" statements. For example, you might have a command of the type –

```
if (condition) goto :label
```

Following is an example of how the goto statement can be used.

## Example

```
@echo off
SET /A a = 5

if %a%==5 goto :labela
if %a%==10 goto :labelb

:labela
echo "The value of a is 5"

exit /b 0

:labelb
echo "The value of a is 10"
```

The key thing to note about the above program is –

The code statements for the label should be on the next line after the declaration of the label.

You can define multiple goto statements and their corresponding labels in a batch file.

The label declarations can be anywhere in the file.

## Output

The above command produces the following output.

```
"The value of a is 5"
```

# Batch Script - Operators

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations.

In batch script, the following types of operators are possible.

- Arithmetic operators

- Relational operators

- Logical operators

- Assignment operators

- Bitwise operators

## Arithmetic Operators

Batch script language supports the normal Arithmetic operators as any language. Following are the Arithmetic operators available.

Show Example

Operator	Description	Example
+	Addition of two operands	1 + 2 will give 3
-	Subtracts second operand from the first	2 - 1 will give 1
*	Multiplication of both operands	2 * 2 will give 4
/	Division of the numerator by the denominator	3 / 2 will give 1.5
%	Modulus operator and remainder of after an integer/float division	3 % 2 will give 1

## Relational Operators

Relational operators allow of the comparison of objects. Below are the relational operators available.

Show Example

Operator	Description	Example
EQU	Tests the equality between two objects	2 EQU 2 will give true
NEQ	Tests the difference between two objects	3 NEQ 2 will give true
LSS	Checks to see if the left object is less than the right operand	2 LSS 3 will give true
LEQ	Checks to see if the left object is less than or equal to the right operand	2 LEQ 3 will give true
GTR	Checks to see if the left object is greater than the right operand	3 GTR 2 will give true
GEQ	Checks to see if the left object is greater than or equal to the right operand	3 GEQ 2 will give true

## Logical Operators

Logical operators are used to evaluate Boolean expressions. Following are the logical operators available.

The batch language is equipped with a full set of Boolean logic operators like AND, OR, XOR, but only for binary numbers. Neither are there any values for TRUE or FALSE. The only logical operator available for conditions is the NOT operator.

Show Example

<b>Operator</b>	<b>Description</b>
AND	This is the logical “and” operator
OR	This is the logical “or” operator
NOT	This is the logical “not” operator

## Assignment Operators

Batch Script language also provides assignment operators. Following are the assignment operators available.

Show Example

<b>Operator</b>	<b>Description</b>	<b>Example</b>
<code>+=</code>	This adds right operand to the left operand and assigns the result to left operand	Set /A a = 5 <code>a += 3</code> Output will be 8
<code>-=</code>	This subtracts the right operand from the left operand and assigns the result to the left operand	Set /A a = 5 <code>a -= 3</code> Output will be 2
<code>*=</code>	This multiplies the right operand with the left operand and assigns the result to the left operand	Set /A a = 5 <code>a *= 3</code> Output will be 15
<code>/=</code>	This divides the left operand with the right operand and assigns the result to the left operand	Set /A a = 6 <code>a/ = 3</code> Output will be 2
<code>%=</code>	This takes modulus using two operands and assigns the result to the left operand	Set /A a = 5 <code>a% = 3</code> Output will be 2

## Bitwise Operators

Bitwise operators are also possible in batch script. Following are the operators available.

#### Show Example

Operator	Description
&	This is the bitwise "and" operator
	This is the bitwise "or" operator
^	This is the bitwise "xor" or Exclusive or operator

Following is the truth table showcasing these operators.

p	q	p & q	p   q	p ^ q
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

## Batch Script - DATE and TIME

The date and time in DOS Scripting have the following two basic commands for retrieving the date and time of the system.

### DATE

This command gets the system date.

#### Syntax

```
DATE
```

#### Example

```
@echo off  
echo %DATE%
```

#### Output

The current date will be displayed in the command prompt. For example,

```
Mon 12/28/2015
```

### TIME

This command sets or displays the time.

## Syntax

```
TIME
```

## Example

```
@echo off  
echo %TIME%
```

## Output

The current system time will be displayed. For example,

```
22:06:52.87
```

Following are some implementations which can be used to get the date and time in different formats.

## Date in Format Year-Month-Day

### Example

```
@echo off  
echo/Today is: %year%-%month%-%day%  
goto :EOF  
setlocal ENABLEEXTENSIONS  
set t = 2&if "%date%" LSS "A" set t = 1  
  
for /f "skip=1 tokens = 2-4 delims = (-)" %%a in ('echo/^|date') do (  
    for /f "tokens = %t%-4 delims=-./ " %%d in ('date/t') do (  
        set %%a=%%%d&set %%b=%%%e&set %%c=%%%f))  
endlocal&set %1=%yy%&set %2=%mm%&set %3=%dd%&goto :EOF
```

## Output

The above command produces the following output.

```
Today is: 2015-12-30
```

## Batch Script - Input / Output

There are three universal “files” for keyboard input, printing text on the screen and printing errors on the screen. The “Standard In” file, known as **stdin**, contains the input to the program/script. The “Standard Out” file, known as **stdout**, is used to write output for display on the screen. Finally, the “Standard Err” file, known as **stderr**, contains any error messages for display on the screen.

Each of these three standard files, otherwise known as the standard streams, are referenced using the numbers 0, 1, and 2. Stdin is file 0, stdout is file 1, and stderr is file 2.

## Redirecting Output (Stdout and Stderr)

One common practice in batch files is sending the output of a program to a log file. The > operator sends, or redirects, stdout or stderr to another file. The following example shows how this can be done.

```
Dir C:\ > list.txt
```

In the above example, the **stdout** of the command Dir C:\ is redirected to the file list.txt.

If you append the number 2 to the redirection filter, then it would redirect the **stderr** to the file lists.txt.

```
Dir C:\ 2> list.txt
```

One can even combine the **stdout** and **stderr** streams using the file number and the '&' prefix. Following is an example.

```
DIR C:\ > lists.txt 2>&1
```

## Suppressing Program Output

The pseudo file NUL is used to discard any output from a program. The following example shows that the output of the command DIR is discarded by sending the output to NUL.

```
Dir C:\ > NUL
```

## Stdin

To work with the Stdin, you have to use a workaround to achieve this. This can be done by redirecting the command prompt's own stdin, called CON.

The following example shows how you can redirect the output to a file called lists.txt. After you execute the below command, the command prompt will take all the input entered by user till it gets an EOF character. Later, it sends all the input to the file lists.txt.

```
TYPE CON > lists.txt
```

## Batch Script - Return Code

By default when a command line execution is completed it should either return zero when execution succeeds or non-zero when execution fails. When a batch script returns a non-zero value after the execution fails, the non-zero value will indicate what is the error number. We will then use the error number to determine what the error is about and resolve it accordingly.

Following are the common exit code and their description.

Error Code	Description
------------	-------------

0	Program successfully completed.
1	Incorrect function. Indicates that Action has attempted to execute non-recognized command in Windows command prompt cmd.exe.
2	The system cannot find the file specified. Indicates that the file cannot be found in specified location.
3	The system cannot find the path specified. Indicates that the specified path cannot be found.
5	Access is denied. Indicates that user has no access right to specified resource.
9009 0x2331	Program is not recognized as an internal or external command, operable program or batch file. Indicates that command, application name or path has been misspelled when configuring the Action.
221225495 0xC0000017 -1073741801	Not enough virtual memory is available.  It indicates that Windows has run out of memory.
3221225786 0xC000013A -1073741510	The application terminated as a result of a CTRL+C. Indicates that the application has been terminated either by the user's keyboard input CTRL+C or CTRL+Break or closing command prompt window.
3221225794 0xC0000142 -1073741502	The application failed to initialize properly. Indicates that the application has been launched on a Desktop to which the current user has no access rights. Another possible cause is that either gdi32.dll or user32.dll has failed to initialize.

## Error Level

The environmental variable %ERRORLEVEL% contains the return code of the last executed program or script.

By default, the way to check for the ERRORLEVEL is via the following code.

### Syntax

```
IF %ERRORLEVEL% NEQ 0 (
    DO_Something
)
```

It is common to use the command EXIT /B %ERRORLEVEL% at the end of the batch file to return the error codes from the batch file.

EXIT /B at the end of the batch file will stop execution of a batch file.

Use EXIT /B < exitcodes > at the end of the batch file to return custom return codes.

Environment variable %ERRORLEVEL% contains the latest errorlevel in the batch file, which is the latest error codes from the last command executed. In the batch file, it is always a good practice to use environment variables instead of constant values, since the same variable get expanded to different values on different computers.

Let's look at a quick example on how to check for error codes from a batch file.

## Example

Let's assume we have a batch file called Find.cmd which has the following code. In the code, we have clearly mentioned that we if don't find the file called lists.txt then we should set the errorlevel to 7. Similarly, if we see that the variable userprofile is not defined then we should set the errorlevel code to 9.

```
if not exist c:\lists.txt exit 7  
if not defined userprofile exit 9  
exit 0
```

Let's assume we have another file called App.cmd that calls Find.cmd first. Now, if the Find.cmd returns an error wherein it sets the errorlevel to greater than 0 then it would exit the program. In the following batch file, after calling the Find.cmd find, it actually checks to see if the errorlevel is greater than 0.

```
Call Find.cmd  
  
if errorlevel gtr 0 exit  
echo "Successful completion"
```

## Output

In the above program, we can have the following scenarios as the output –

If the file c:\lists.txt does not exist, then nothing will be displayed in the console output.

If the variable userprofile does not exist, then nothing will be displayed in the console output.

If both of the above condition passes then the string "Successful completion" will be displayed in the command prompt.

## Loops

In the decision making chapter, we have seen statements which have been executed one after the other in a sequential manner. Additionally, implementations can also be done in Batch Script to alter the flow of control in a program's logic. They are then classified into

flow of control statements.

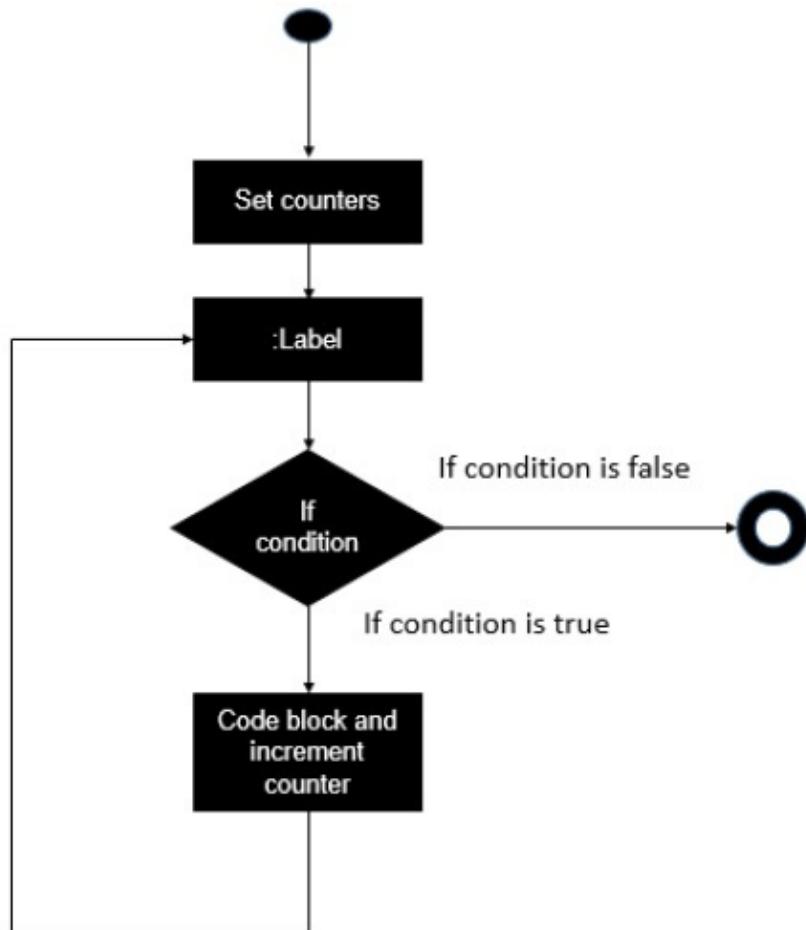
S.No	Loops & Description
1	<b>While Statement Implementation</b> There is no direct while statement available in Batch Script but we can do an implementation of this loop very easily by using the if statement and labels.
2	<b>For Statement - List Implementations</b> The "FOR" construct offers looping capabilities for batch files. Following is the common construct of the 'for' statement for working with a list of values.
3	<b>Looping through Ranges</b> The 'for' statement also has the ability to move through a range of values. Following is the general form of the statement.
4	<b>Classic for Loop Implementation</b> Following is the classic 'for' statement which is available in most programming languages.

# While Statement Implementation

Advertisements

There is no direct while statement available in Batch Script but we can do an implementation of this loop very easily by using the if statement and labels.

The following diagram shows the diagrammatic explanation of this loop.



The first part of the while implementation is to set the counters which will be used to control the evaluation of the 'if' condition. We then define our label which will be used to embody the entire code for the while loop implementation. The 'if' condition evaluates an expression. If the expression evaluates to true, the code block is executed. If the condition evaluates to false then the loop is exited. When the code block is executed, it will return back to the label statement for execution again.

Following is the syntax of the general implementation of the while statement.

## Syntax

```
Set counters
:label
If (expression) (
    Do_something
    Increment counter
    Go back to :label
)
```

The entire code for the while implementation is placed inside of a label.

The counter variables must be set or initialized before the while loop implementation starts.

The expression for the while condition is done using the 'if' statement. If the expression evaluates to true then the relevant code inside the 'if' loop is executed.

A counter needs to be properly incremented inside of 'if' statement so that the while implementation can terminate at some point in time.

Finally, we will go back to our label so that we can evaluate our 'if' statement again.

Following is an example of a while loop statement.

## Example

```
@echo off
SET /A "index = 1"
SET /A "count = 5"
:while
if %index% leq %count% (
    echo The value of index is %index%
    SET /A "index = index + 1"
    goto :while
)
```

In the above example, we are first initializing the value of an index integer variable to 1. Then our condition in the 'if' loop is that we are evaluating the condition of the expression to be that index should it be less than the value of the count variable. Till the value of index is less than 5, we will print the value of index and then increment the value of index.

## Output

The above command produces the following output.

```
The value of index is 1
The value of index is 2
```

The value of index is 3

The value of index is 4

The value of index is 5

# For Statement List Implementations

Advertisements

The "FOR" construct offers looping capabilities for batch files. Following is the common construct of the 'for' statement for working with a list of values.

## Syntax

```
FOR %%variable IN list DO do_something
```

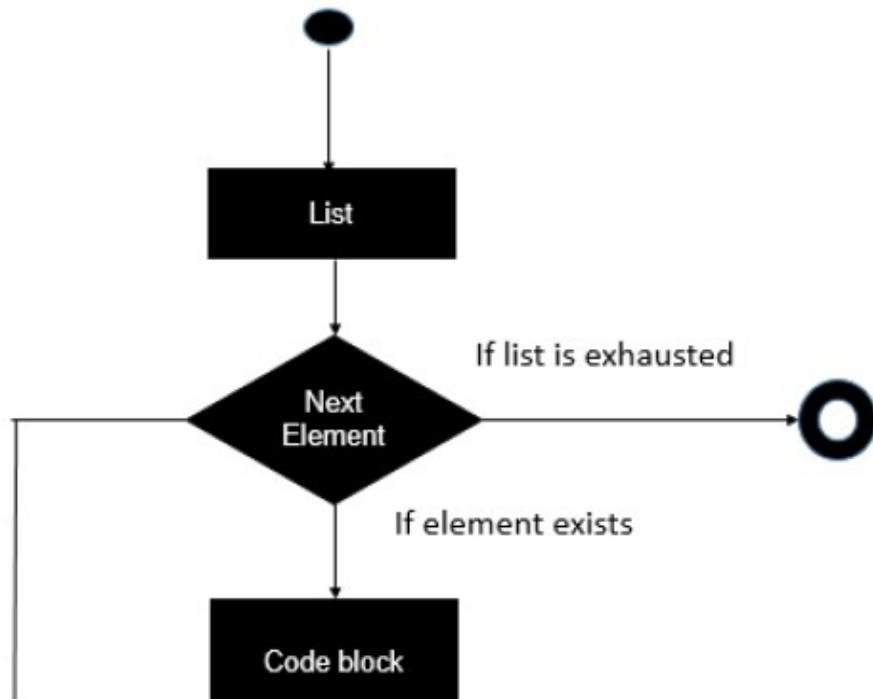
The classic 'for' statement consists of the following parts –

Variable declaration – This step is executed only once for the entire loop and used to declare any variables which will be used within the loop. In Batch Script, the variable declaration is done with the %% at the beginning of the variable name.

List – This will be the list of values for which the 'for' statement should be executed.

The do\_something code block is what needs to be executed for each iteration for the list of values.

The following diagram shows the diagrammatic explanation of this loop.





Following is an example of how the 'goto' statement can be used.

## Example

```
@echo off  
FOR %%F IN (1 2 3 4 5) DO echo %%F
```

The key thing to note about the above program is –

The variable declaration is done with the %% sign at the beginning of the variable name.

The list of values is defined after the IN clause.

The do\_something code is defined after the echo command. Thus for each value in the list, the echo command will be executed.

## Output

The above program produces the following output.

```
1  
2  
3  
4  
5
```

# Batch Script - Looping through Ranges

Advertisements

The 'for' statement also has the ability to move through a range of values. Following is the general form of the statement.

## Syntax

```
FOR /L %%variable IN (lowerlimit,Increment,Upperlimit) DO do_something
```

Where

The /L switch is used to denote that the loop is used for iterating through ranges.

Variable declaration – This step is executed only once for the entire loop and used to declare any variables which will be used within the loop. In Batch Script, the variable declaration is done with the %% at the beginning of the variable name.

The IN list contains of 3 values. The lowerlimit, the increment, and the upperlimit. So, the loop would start with the lowerlimit and move to the upperlimit value, iterating each time by the Increment value.

The do\_something code block is what needs to be executed for each iteration.

Following is an example of how the looping through ranges can be carried out.

## Example

```
@ECHO OFF
FOR /L %%X IN (0,1,5) DO ECHO %%X
```

## Output

The above program produces the following output.

```
0
1
2
3
4
```

# Classic for Loop Implementation

Advertisements

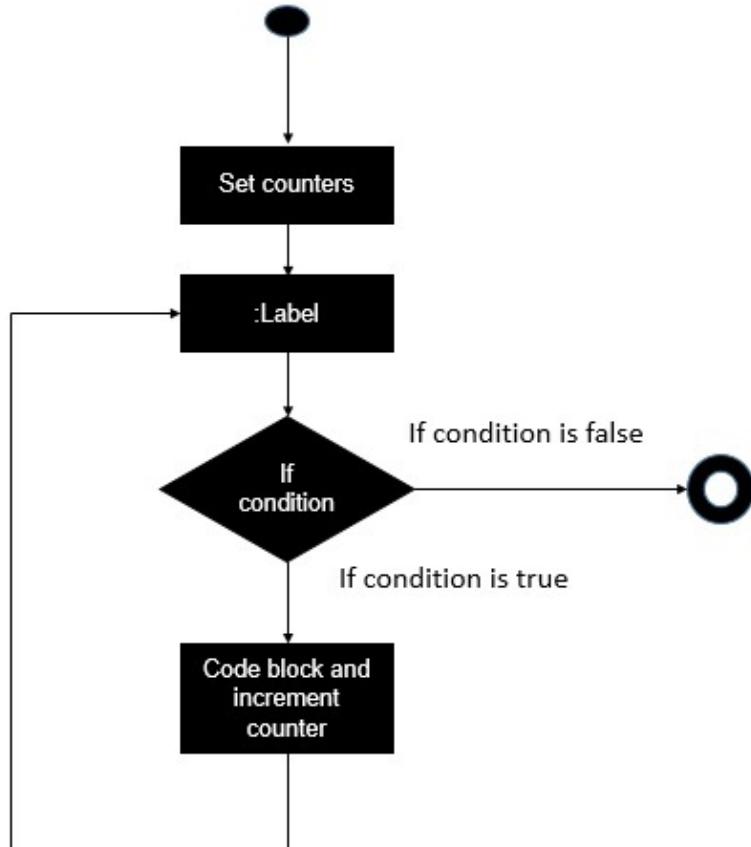
Following is the classic 'for' statement which is available in most programming languages.

## Syntax

```
for(variable declaration;expression;Increment) {
    statement #1
    statement #2
    ...
}
```

The Batch Script language does not have a direct 'for' statement which is similar to the above syntax, but one can still do an implementation of the classic 'for' loop statement using if statements and labels.

Following is the general flow of the classic 'for' loop statement.



Let's look at the general syntax implementation of the classic for loop in batch scripting.

```
Set counter
:label

If (expression) exit loop
Do_something
Increment counter
Go back to :label
```

The entire code for the 'for' implementation is placed inside of a label.

The counters variables must be set or initialized before the 'for' loop implementation starts.

The expression for the 'for' loop is done using the 'if' statement. If the expression evaluates to be true then an exit is executed to come out of the loop.

A counter needs to be properly incremented inside of the 'if' statement so that the 'for' implementation can continue if the expression evaluation is false.

Finally, we will go back to our label so that we can evaluate our 'if' statement again.

Following is an example of how to carry out the implementation of the classic 'for' loop statement.

## Example

```
@echo off
SET /A i = 1
:loop

IF %i%==5 GOTO END
echo The value of i is %i%
SET /a i=%i%+1
GOTO :LOOP
:END
```

## Output

The above command produces the following output.

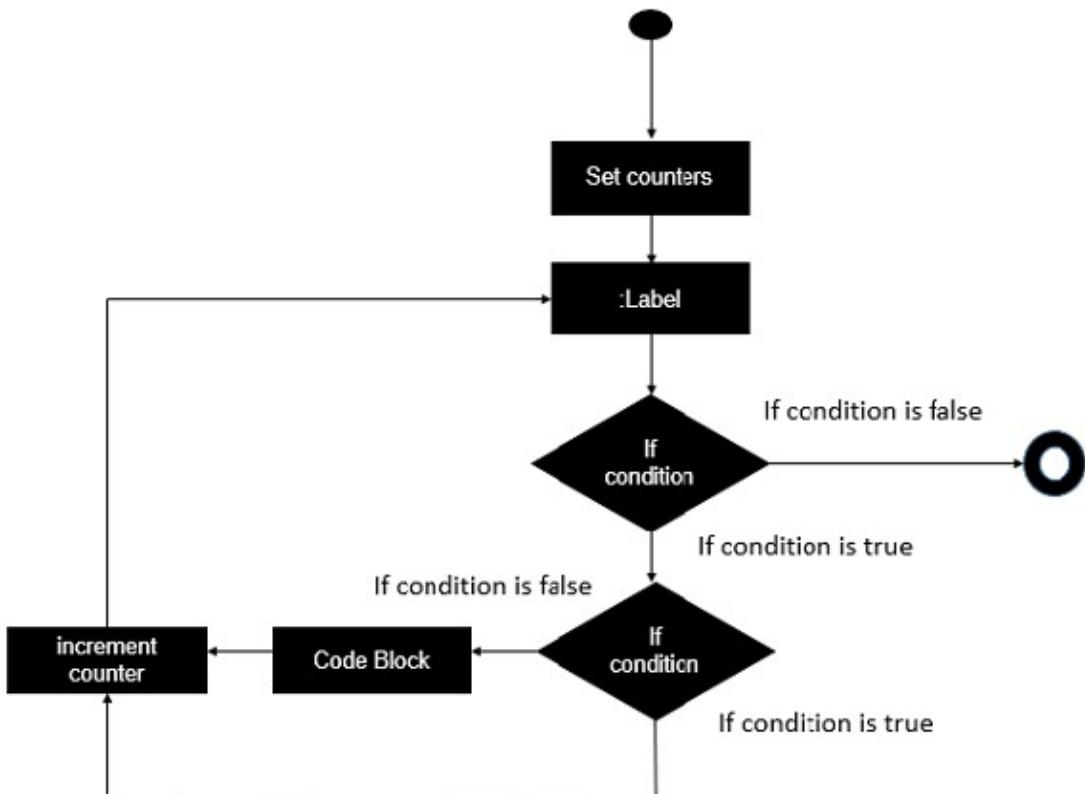
```
The value of i is 1
The value of i is 2
The value of i is 3
The value of i is 4
```

# Break Statement Implementation

Advertisements

The break statement is used to alter the flow of control inside loops within any programming language. The break statement is normally used in looping constructs and is used to cause immediate termination of the innermost enclosing loop.

The Batch Script language does not have a direct 'for' statement which does a break but this can be implemented by using labels. The following diagram shows the diagrammatic explanation of the break statement implementation in Batch Script.



The key thing to note about the above implementation is the involvement of two 'if' conditions. The second 'if' condition is used to control when the break is implemented. If the second 'if' condition is evaluated to be true, then the code block is not executed and the counter is directly implemented.

Following is an example of how to carry out the implementation of the break statement.

## Example

```
@echo off
SET /A "index=1"
SET /A "count=5"
:while
if %index% leq %count% (
    if %index%==2 goto :Increment
    echo The value of index is %index%
:Increment
    SET /A "index=index + 1"
    goto :while
)
```

The key thing to note about the above program is the addition of a label called :Increment. When the value of index reaches 2, we want to skip the statement which echoes its value to the command prompt and directly just increment the value of index.

## Output

The above command produces the following output.

```
The value of index is 1
The value of index is 3
The value of index is 4
The value of index is 5
```

# Looping through Command Line Arguments

The 'for' statement can also be used for checking command line arguments. The following example shows how the 'for' statement can be used to loop through the command line arguments.

## Example

```
@ECHO OFF  
:Loop  
  
IF "%1""="" GOTO completed  
FOR %%F IN (%1) DO echo %%F  
SHIFT  
GOTO Loop  
:completed
```

## Output

Let's assume that our above code is stored in a file called Test.bat. The above command will produce the following output if the batch file passes the command line arguments of 1,2 and 3 as Test.bat 1 2 3.

```
1  
2  
3
```

S.No	Loops & Description
1	<p><b>Break Statement Implementation</b></p> <p>The break statement is used to alter the flow of control inside loops within any programming language. The break statement is normally used in looping constructs and is used to cause immediate termination of the innermost enclosing loop.</p>

## Batch Script - Functions

A function is a set of statements organized together to perform a specific task. In batch scripts, a similar approach is adopted to group logical statements together to form a function.

As like any other languages, functions in Batch Script follows the same procedure –

**Function Declaration** – It tells the compiler about a function's name, return type, and parameters.

**Function Definition** – It provides the actual body of the function.

### Function Definition

In Batch Script, a function is defined by using the label statement. When a function is newly defined, it may take one or several values as input 'parameters' to the function, process the functions in the main body, and pass back the values to the functions as output 'return types'.

Every function has a function name, which describes the task that the function performs. To use a function, you "call" that function with its name and pass its input values (known as arguments) that matches the types of the function's parameters.

Following is the syntax of a simple function.

```
:function_name
Do_something
EXIT /B 0
```

The function\_name is the name given to the function which should have some meaning to match what the function actually does.

The EXIT statement is used to ensure that the function exits properly.

Following is an example of a simple function.

### Example

```

:Display
SET /A index=2
echo The value of index is %index%
EXIT /B 0

```

S.No	Functions & Description
1	<b>Calling a Function</b> A function is called in Batch Script by using the call command.
2	<b>Functions with Parameters</b> Functions can work with parameters by simply passing them when a call is made to the function.
3	<b>Functions with Return Values</b> Functions can work with return values by simply passing variables names
4	<b>Local Variables in Functions</b> Local variables in functions can be used to avoid name conflicts and keep variable changes local to the function.
5	<b>Recursive Functions</b> The ability to completely encapsulate the body of a function by keeping variable changes local to the function and invisible to the caller.
6	<b>File I/O</b> In Batch Script, it is possible to perform the normal file I/O operations that would be expected in any programming language.
7	<b>Creating Files</b> The creation of a new file is done with the help of the redirection filter >. This filter can be used to redirect any output to a file.
8	<b>Writing to Files</b> Content writing to files is also done with the help of the redirection filter >. This filter can be used to redirect any output to a file.
9	<b>Appending to Files</b> Content writing to files is also done with the help of the double redirection filter >>. This filter can be used to append any output to a file.
10	<b>Reading from Files</b> Reading of files in a batch script is done via using the FOR loop command to go through each line which is defined in the file that needs to be read.

	<b>Deleting Files</b> For deleting files, Batch Script provides the DEL command.
11	<b>Renaming Files</b> For renaming files, Batch Script provides the REN or RENAME command.
12	<b>Moving Files</b> For moving files, Batch Script provides the MOVE command.
13	<b>Batch Files – Pipes</b> The pipe operator ( ) takes the output (by default, STDOUT) of one command and directs it into the input (by default, STDIN) of another command.
14	<b>Batch Files – Inputs</b> When a batch file is run, it gives you the option to pass in command line parameters which can then be read within the program for further processing.
15	<b>Using the SHIFT Operator</b> One of the limitations of command line arguments is that it can accept only arguments till %9. Let's take an example of this limitation.
16	<b>Folders</b> In Batch Script, it is possible to perform the normal folder based operations that would be expected in any programming language.
17	<b>Creating Folders</b> The creation of a folder is done with the assistance of the MD (Make directory) command.
18	<b>Listing Folder Contents</b> The listing of folder contents can be done with the dir command. This command allows you to see the available files and directories in the current directory.
19	<b>Deleting Folders</b> For deleting folders, Batch Scripting provides the DEL command.
20	<b>Renaming Folders</b> For renaming folders, Batch Script provides the REN or RENAME command.
21	<b>Moving Folders</b> For moving folders, Batch Script provides the MOVE command.

# Batch Script - Calling a Function

Advertisements

A function is called in Batch Script by using the call command. Following is the syntax.

## Syntax

```
call :function_name
```

Following example shows how a function can be called from the main program.

## Example

```
@echo off
SETLOCAL
CALL :Display
EXIT /B %ERRORLEVEL%
:Display
SET /A index=2
echo The value of index is %index%
EXIT /B 0
```

One key thing to note when defining the main program is to ensure that the statement EXIT /B %ERRORLEVEL% is put in the main program to separate the code of the main program from the function.

## Output

The above command produces the following output.

```
The value of index is 2
```

# Batch Script - Functions with Parameters

Advertisements

Functions can work with parameters by simply passing them when a call is made to the function.

## Syntax

```
Call :function_name parameter1, parameter2... parameterN
```

The parameters can then be accessed from within the function by using the tilde (~) character along with the positional number of the parameter.

Following example shows how a function can be called with parameters.

## Example

```
@echo off
SETLOCAL
CALL :Display 5 , 10
EXIT /B %ERRORLEVEL%
:Display
echo The value of parameter 1 is %~1
echo The value of parameter 2 is %~2
EXIT /B 0
```

As seen in the above example, ~1 is used to access the first parameter sent to the function, similarly ~2 is used to access the second parameter.

## Output

The above command produces the following output.

```
The value of parameter 1 is 5
The value of parameter 2 is 10
```

# Batch Script - Functions with Return Values

Advertisements

Functions can work with return values by simply passing variables names which will hold the return values when a call is made to the function as shown below

## Syntax

```
Call :function_name value1, value2... valuen
```

The return values are set in the function using the set command and the tilde(~) character along with the positional number of the parameter.

Following example shows how a function can be called with return values.

## Example

```
@echo off
SETLOCAL
CALL :SetValue value1,value2
echo %value1%
echo %value2%
EXIT /B %ERRORLEVEL%
:SetValue
set "%~1 = 5"
set "%~2 = 10"
EXIT /B 0
```

## Output

The above command produces the following output.

```
5
10
```



# Batch Script - Local Variables in Functions

Advertisements

Local variables in functions can be used to avoid name conflicts and keep variable changes local to the function. The SETLOCAL command is first used to ensure the command processor takes a backup of all environment variables. The variables can be restored by calling ENDLOCAL command. Changes made in between are local to the current batch script. ENDLOCAL is automatically called when the end of the batch file is reached, i.e. by calling GOTO:EOF.

Localizing variables with SETLOCAL allows using variable names within a function freely without worrying about name conflicts with variables used outside the function.

Following example shows how local variables can be used in functions.

## Example

```
@echo off
set str = Outer
echo %str%
CALL :SetValue str
echo %str%
EXIT /B %ERRORLEVEL%
:SetValue
SETLOCAL
set str = Inner
set "%~1 = %str%"
ENDLOCAL
EXIT /B 0
```

## Output

In the above program, the variable 'str' is being localized in the function SetValue. Thus even though the str value is being returned back to the main function, the value of str in the main function will not be replaced by the value being returned from the function.

The above command produces the following output.

```
Outer
Outer
```

# Batch Script - Recursive Functions

Advertisements

The ability to completely encapsulate the body of a function by keeping variable changes local to the function and invisible to the caller. We can now have the ability to call a function recursively making sure each level of recursion works with its own set of variables even though variable names are being reused.

Following example shows how recursive functions can be used.

## Example

The example shows how to calculate a Fibonacci number recursively. The recursion stops when the Fibonacci algorithm reaches a number greater or equal to a given input number. The example starts with the numbers 0 and 1, the :myFibo function calls itself recursively to calculate the next Fibonacci number until it finds the Fibonacci number greater or equal to 1000000000.

The first argument of the myFibo function is the name of the variable to store the output in. This variable must be initialized to the Fibonacci number to start with and will be used as current Fibonacci number when calling the function and will be set to the subsequent Fibonacci number when the function returns.

```
@echo off
set "fst = 0"
set "fib = 1"
set "limit = 1000000000"
call:myFibo fib,%fst%,%limit%
echo.The next Fibonacci number greater or equal %limit% is %fib%.
echo.&pause&goto:eof
:myFibo -- calculate recursively
:myFibo -- calculate recursively the next Fibonacci number greater or equal to a limit
SETLOCAL
set /a "Number1 = %~1"
set /a "Number2 = %~2"
set /a "Limit = %~3"
set /a "NumberN = Number1 + Number2"

if /i %NumberN% LSS %Limit% call:myFibo NumberN,%Number1%,%Limit%
(ENDLOCAL
    IF "%~1" NEQ "" SET "%~1 = %NumberN%"
)goto:eof
```

## Output

The above command produces the following output.

```
The next Fibonacci number greater or equal 1000000000 is 1134903170.
```



# Batch Script - File I/O

Advertisements

In Batch Script, it is possible to perform the normal file I/O operations that would be expected in any programming language.

Following are some of the operations that can be performed on files.

Creating files

Reading files

Writing to files

Deleting files

Moving files

Renaming files



# Batch Script - Creating Files

Advertisements

The creation of a new file is done with the help of the redirection filter >. This filter can be used to redirect any output to a file. Following is a simple example of how to create a file using the redirection command.

## Example

```
@echo off  
echo "Hello">>C:\new.txt
```

## Output

If the file new.txt is not present in C:\, then it will be created with the help of the above command.

# Batch Script - Writing to Files

Advertisements

Content writing to files is also done with the help of the redirection filter >. This filter can be used to redirect any output to a file. Following is a simple example of how to create a file using the redirection command to write data to files.

## Example

```
@echo off  
dir C:\>C:\new.txt
```

The above code snippet first uses the DIR command to get the directory listing of the entire C:\ . It then takes that output and with the help of the redirection command sends it to the file new.txt.

## Output

If you open up the file new.txt on your C drive, you will get the contents of your C drive in this file. Following is a sample output.

```
Volume in drive C is Windows8_OS  
Volume Serial Number is E41C-6F43  
  
Directory of C:\  
  
12/22/2015 09:02 PM    <DIR>        01 - Music  
06/14/2015 10:31 AM    <DIR>        02 - Videos  
09/12/2015 06:23 AM    <DIR>        03 - Pictures  
12/17/2015 12:19 AM    <DIR>        04 - Software  
12/15/2015 11:06 PM    <DIR>        05 - Studies  
12/20/2014 09:09 AM    <DIR>        06 - Future  
12/20/2014 09:07 AM    <DIR>        07 - Fitness  
09/19/2015 09:56 AM    <DIR>        08 - Tracking  
10/19/2015 10:28 PM    <DIR>        09 - Misc
```

# Batch Script - Appending to Files

Advertisements

Content writing to files is also done with the help of the double redirection filter >>. This filter can be used to append any output to a file. Following is a simple example of how to create a file using the redirection command to append data to files.

## Example

```
@echo off
echo "This is the directory listing of C:\ Drive">>C:\new.txt
dir C:\>>C:\new.txt
```

In the above example, you can see that the first echo command is used to create the file using the single redirection command whereas the DIR command is outputted to the file using the double redirection filter.

## Output

If you open the file new.txt on your C drive, you will get the contents of your C drive in this file plus the string "This is the directory listing of C:\ Drive" . Following is a sample output.

```
"This is the directory listing of C:\ Drive"
Volume in drive C is Windows8_OS
Volume Serial Number is E41C-6F43

Directory of C:\

12/22/2015 09:02 PM    <DIR>        01 - Music
06/14/2015 10:31 AM    <DIR>        02 - Videos
09/12/2015 06:23 AM    <DIR>        03 - Pictures
12/17/2015 12:19 AM    <DIR>        04 - Software
12/15/2015 11:06 PM    <DIR>        05 - Studies
12/20/2014 09:09 AM    <DIR>        06 - Future
12/20/2014 09:07 AM    <DIR>        07 - Fitness
09/19/2015 09:56 AM    <DIR>        08 - Tracking
10/19/2015 10:28 PM    <DIR>        09 - Misc
```

# Batch Script - Reading from Files

Advertisements

Reading of files in a Batch Script is done via using the FOR loop command to go through each line which is defined in the file that needs to be read. Since there is no direct command to read text from a file into a variable, the 'for' loop needs to be used to serve this purpose.

Let's look at an example on how this can be achieved.

## Example

```
@echo off
FOR /F "tokens=* delims=" %%x in (new.txt) DO echo %%x
```

The delims parameter is used to break up the text in the file into different tokens or words. Each word or token is then stored in the variable x. For each word which is read from the file, an echo is done to print the word to the console output.

## Output

If you consider the new.txt file which has been considered in previous examples, you might get the following output when the above program is run.

```
"This is the directory listing of C:\ Drive"
Volume in drive C is Windows8_OS
Volume Serial Number is E41C-6F43

Directory of C:\  

12/22/2015 09:02 PM <DIR> 01 - Music
06/14/2015 10:31 AM <DIR> 02 - Videos
09/12/2015 06:23 AM <DIR> 03 - Pictures
12/17/2015 12:19 AM <DIR> 04 - Software
12/15/2015 11:06 PM <DIR> 05 - Studies
12/20/2014 09:09 AM <DIR> 06 - Future
12/20/2014 09:07 AM <DIR> 07 - Fitness
09/19/2015 09:56 AM <DIR> 08 - Tracking
```



# Batch Script - Deleting Files

Advertisements

For deleting files, Batch Script provides the DEL command.

## Syntax

```
DEL [/P] [/F] [/S] [/Q] [/A[[[:]]attributes]] names
```

Following are the description of the options which can be presented to the DEL command.

S.No.	Options & Description
1.	<b>Names</b> Specifies a list of one or more files or directories. Wildcards may be used to delete multiple files. If a directory is specified, all files within the directory will be deleted
2.	<b>/P</b> Prompts for confirmation before deleting each file.
3.	<b>/F</b> Force deletes of read-only files.
4.	<b>/S</b> Deletes specified files from all subdirectories.
5.	<b>/Q</b> Quiet mode, do not ask if ok to delete on global wildcard.
6.	<b>/A</b> Selects files to delete based on attributes.
7.	<b>attributes</b>

R Read-only files, S System files, H Hidden files, A Files ready for archiving -  
Prefix meaning not

Following examples show how the DEL command can be used.

## Examples

```
del test.bat
```

The above command will delete the file test.bat in the current directory, if the file exists.

```
del c:\test.bat
```

The above command will delete the file C:\test.bat in the current directory, if the file exists.

```
del c:\*.bat
```

The \* (asterisks) is a wild character. \*.bat indicates that you would like to delete all bat files in the c:\directory.

```
del c:\?est.tmp
```

The ? (question mark) is a single wild character for one letter. The use of this command in the above example will delete any file ending with "est.tmp", such as pest.tmp or test.tmp.



# Batch Script - Renaming Files

Advertisements

For renaming files, Batch Script provides the REN or RENAME command.

## Syntax

```
RENAME [drive:][path][directoryname1 | filename1] [directoryname2 | filename2]
```

Let's take a look at some examples of renaming files.

## Examples

```
rename *.txt *.bak
```

The above command will rename all text files to files with .bak extension.

```
rename "TESTA.txt" "TESTB.txt"
```

The above command will rename the file TESTA.txt to TESTB.txt.



# Batch Script - Moving Files

Advertisements

For moving files, Batch Script provides the MOVE command.

## Syntax

```
MOVE [/Y | /-Y] [drive:][path]filename1[,...] destination
```

Following are the description of the options which can be presented to the DEL command.

S.No.	Options & Description
1.	<b>[drive:][path]filename1</b> Specifies the location and name of the file or files you want to move
2.	<b>destination</b> Specifies the new location of the file. Destination can consist of a drive letter and colon, a directory name, or a combination. If you are moving only one file, you can also include a filename if you want to rename the file when you move it.
3.	<b>[drive:][path]dirname1</b> Specifies the directory you want to rename.
4.	<b>dirname2</b> Specifies the new name of the directory.
5.	<b>/Y</b> Suppresses prompting to confirm you want to overwrite an existing destination file.
6.	<b>/-Y</b> Causes prompting to confirm you want to overwrite an existing destination file.

Let's look at some examples of renaming files.

## Examples

```
move c:\windows\temp\*.* c:\temp
```

The above command will move the files of c:\windows\temp to the temp directory in root.

```
move new.txt, test.txt c:\example
```

The above command will move the files new.txt and test.txt into the c:\example folder.



# Batch Script - Files Pipes

Advertisements

The pipe operator (|) takes the output (by default, STDOUT) of one command and directs it into the input (by default, STDIN) of another command. For example, the following command sorts the contents of the directory C:\

```
dir C:\ | sort
```

In this example, both commands start simultaneously, but then the sort command pauses until it receives the dir command's output. The sort command uses the dir command's output as its input, and then sends its output to handle 1 (that is, STDOUT).

Following is another example of the pipe command. In this example, the contents of the file C:\new.txt are sent to the sort command through the pipe filter.

```
@echo off  
TYPE C:\new.txt | sort
```

## Combining Commands with Redirection Operators

Usually, the pipe operator is used along with the redirection operator to provide useful functionality when it comes to working with pipe commands.

For example, the below command will first take all the files defined in C:\, then using the pipe command, will find all the files with the .txt extension. It will then take this output and print it to the file AllText.txt.

```
dir C:\ | find "txt" > AllText.txt
```

## Using Multiple Pipe Commands

To use more than one filter in the same command, separate the filters with a pipe (|). For example, the following command searches every directory on drive C:, finds the file names that include the string "Log", and then displays them in one Command Prompt window at a time –

```
dir c:\ /s /b | find "TXT" | more
```

Following are some examples of how the pipe filter can be used.

## Examples

The following example sends the list of all running tasks using the tasklist command and sends the output to the find command. The find command will then find all processes which are of the type notepad and display them in the command prompt.

```
tasklist | find "notepad"
```

## Output

Following is a sample output.

notepad.exe	1400	Console	1	8,916 K
notepad.exe	4016	Console	1	11,200 K
notepad.exe	1508	Console	1	8,720 K
notepad.exe	4076	Console	1	8,688 K

The following example sends the list of all running tasks using the tasklist command and sends the output to the more command. The more command will then display the lists of running tasks one page at a time.

## Example

```
tasklist | more
```

## Output

Image Name	PID	Session Name	Session#	Mem Usage
System Idle Process	0	Services	0	4 K
System	4	Services	0	276 K
smss.exe	344	Services	0	1,060 K
csrss.exe	524	Services	0	4,188 K
csrss.exe	608	Console	1	58,080 K
wininit.exe	616	Services	0	3,528 K
winlogon.exe	644	Console	1	5,636 K
services.exe	708	Services	0	7,072 K
lsass.exe	716	Services	0	10,228 K
svchost.exe	784	Services	0	10,208 K
svchost.exe	828	Services	0	7,872 K
dwm.exe	912	Console	1	208,316 K
nvvsvc.exe	932	Services	0	6,772 K
nvxdsync.exe	968	Console	1	16,584 K
nvvsvc.exe	976	Console	1	12,780 K
svchost.exe	1008	Services	0	20,340 K
svchost.exe	224	Services	0	39,740 K
svchost.exe	468	Services	0	11,864 K
svchost.exe	860	Services	0	11,184 K

```
svchost.exe           232 Services      0     16,992 K
wlanext.exe          1168 Services     0     12,840 K
-- More --
```

The following example sends the list of all running tasks using the tasklist command and sends the output to the find command. The find command will then find all processes which are of the type notepad and then uses the redirection command to send the content to the file tasklist.txt.

## Example

```
tasklist | find "notepad" > tasklist.txt
```

## Output

If you open the file tasklist.txt, you will get the following sample output.

```
notepad.exe          1400 Console      1     8,916 K
notepad.exe          4016 Console      1    11,200 K
notepad.exe          1508 Console      1     8,720 K
notepad.exe          4076 Console      1     8,688 K
```



# Batch Script - Files Inputs

Advertisements

When a batch file is run, it gives you the option to pass in command line parameters which can then be read within the program for further processing. The batch files parameters can be recalled from within the program using the % operator along with the numeric position of the parameter. Following is how the command line parameters are defined.

%0 is the program name as it was called.

%1 is the first command line parameter.

%2 is the second command line parameter.

So on till %9.

Let's take a look at a simple example of how command line parameters can be used.

## Example

```
@echo off  
echo The first parameter is %1  
echo The second parameter is %2  
echo The third parameter is %3
```

## Output

If the above code is stored in a file called test.bat and the file is run as

```
test.bat 5 10 15
```

then, following will be the output.

```
The first parameter is 5  
The second parameter is 10  
The third parameter is 15
```

# Batch Script - Using the SHIFT Operator

Advertisements

One of the limitations of command line arguments is that it can accept only arguments till %9. Let's take an example of this limitation.

## Example

```
@echo off
echo %1
echo %2
echo %3
echo %4
echo %5
echo %6
echo %7
echo %8
echo %9
echo %10
```

## Output

If the above code is stored in a file called test.bat and the file is run as

```
test.bat a b c d e f g h i j
```

Then following will be the output.

```
a
b
c
d
e
f
g
h
i
a0
```

As you can see from the above output, the final value which should be shown as 'j' is being shown as a0. This is because there is no parameter known as %10.

This limitation can be avoided by using the SHIFT operator. After your batch file handled its

first parameter(s) it could SHIFT them (just insert a line with only the command SHIFT), resulting in %1 getting the value B, %2 getting the value C, etcetera, till %9, which now gets the value J. Continue this process until at least %9 is empty.

Let's look at an example of how to use the SHIFT operator to overcome the limitation of command line arguments.

## Example

```
@ECHO OFF  
:Loop  
  
IF "%1"=="" GOTO Continue  
echo %1%  
SHIFT  
GOTO Loop  
:Continue
```

If the above code is stored in a file called test.bat and the file is run as

```
test.bat a b c d e f g h i j
```

Then following will be the output.

```
a  
b  
c  
d  
e  
f  
h  
i  
j
```

## Note

Some characters in the command line are ignored by batch files, depending on the DOS version, whether they are "escaped" or not, and often depending on their location in the command line –

Commas (",") are replaced by spaces, unless they are part of a string in doublequotes.

Semicolons (";") are replaced by spaces, unless they are part of a string in doublequotes.

"=" characters are sometimes replaced by spaces, not if they are part of a string in doublequotes.

The first forward slash ("/") is replaced by a space only if it immediately follows the command, without a leading space.

Multiple spaces are replaced by a single space, unless they are part of a string in doublequotes.

Tabs are replaced by a single space.

Leading spaces before the first command line argument are ignored.

Trailing spaces after the last command line argument are trimmed.

---



# Batch Script - Folders

Advertisements

In Batch Script, it is possible to perform the normal folder based operations that would be expected in any programming language.

Following are some of the operations that can be performed on folders.

Creating folders

Listing folders

Traversing files in folders

Deleting folders

Renaming folders



# Batch Script - Creating Folders

Advertisements

The creation of a folder is done with the assistance of the MD (Make directory) command.

## Syntax

```
MKDIR [drive:]path  
MD [drive:]path
```

Let's look at some examples on how to use the MD command.

## Examples

```
md test
```

The above command will create a directory called test in your current directory.

```
md C:\test
```

The above command will create a directory called test in the C drive.

```
md "Test A"
```

If there are spaces in the folder name, then the folder name should be given in quotes.

```
mkdir \a\b\c
```

The above command creates directories recursively and is the same as issuing the following set of commands.

```
mkdir \a  
chdir \a  
mkdir b  
chdir b  
mkdir c
```

# Batch Script - Listing Folder Contents

Advertisements

The listing of folder contents can be done with the dir command. This command allows you to see the available files and directories in the current directory. The dir command also shows the last modification date and time, as well as the file size.

## Syntax

```
DIR [drive:][path][filename] [/A[[:]:attributes]] [/B] [/C] [/D] [/L] [/N]
[/O[[:]:sortorder]] [/P] [/Q] [/R] [/S] [/T[[:]:timefield]] [/W] [/X] [/4]
```

S.No.	Options & Description
1.	<b>[drive:][path][filename]</b> Specifies drive, directory, or files to list
2.	<b>/A</b> Displays files with specified attributes.
3.	<b>attributes</b> D - Directories      R - Read-only files H - Hidden files      A - Files ready for archiving S - System files      I - Not content indexed files L - Reparse Points    - Prefix meaning not
4.	<b>/B</b> Uses bare format (no heading information or summary).
5.	<b>/C</b> Displays the thousand separator in file sizes. This is the default. Use /-C to disable the display of the separator.

Let's see some of the examples on how to use the DIR command.

## Examples

```
dir *.exe
```

The above command lists any file that ends with the .exe file extension.

```
dir *.txt *.doc
```

The above command uses multiple filespecs to list any files ending with .txt and .doc in one command.

```
dir /ad
```

Lists only the directories in the current directory. If you need to move into one of the directories listed use the cd command.

```
dir /s
```

Lists the files in the directory that you are in and all sub directories after that directory. If you are at root "C:\>", type this command, this will list to you every file and directory on the C: drive of the computer.

```
dir /p
```

If the directory has lots of files and you cannot read all the files as they scroll by, you can use the above command and it displays all files one page at a time.

```
dir /w
```

If you don't need file information you can use the above command to list only the files and directories going horizontally, taking as little space as needed.

```
dir /s /w /p
```

The above command will list all the files and directories in the current directory and the sub directories, in wide format and one page at a time.



# Batch Script - Deleting Folders

Advertisements

For deleting folders, Batch Script provides the DEL command.

## Syntax

```
DEL [/P] [/F] [/S] [/Q] [/A[[[:]]attributes]] names
```

Following are the description of the options which can be presented to the DEL command.

S.No.	Options & Description
1.	<b>Names</b> Specifies a list of one or more files or directories. Wildcards may be used to delete multiple files. If a directory is specified, all files within the directory will be deleted
2.	<b>/P</b> Prompts for confirmation before deleting each file.
3.	<b>/F</b> Force deletes read-only files.
4.	<b>/S</b> Deletes specified files from all subdirectories.
5.	<b>/Q</b> Quiet mode, do not ask if ok to delete on global wildcard.
6.	<b>/A</b> Selects files to delete based on attributes.
7.	<b>attributes</b>

R - Read-only files, S - System files, H - Hidden files, A - Files ready for archiving  
- Prefix meaning not

Let's look at some examples of how the DEL command can be used for folders.

## Examples

```
del Example
```

The above command will delete the folder called Example in the current working directory.

```
del C:\Example
```

The above command will delete the folder called Example in C drive.

```
Del Example1 , Example2
```

The above command will delete the folder called Example1 and Example2 in the current working directory.



# Batch Script - Renaming Folders

Advertisements

For renaming folders, Batch Script provides the REN or RENAME command.

## Syntax

```
RENAME [drive:][path][directoryname1 | filename1] [directoryname2 | filename2]
```

Let's look at some examples of renaming folders.

## Examples

```
ren Example Example1
```

The above command will rename the folder called Example in the current working directory to Example1.

```
del C:\Example Example1
```

The above command will rename the folder called Example in C Drive to Example1.

# Batch Script - Moving Folders

Advertisements

For moving folders, Batch Script provides the MOVE command.

## Syntax

```
MOVE [/Y | /-Y] [drive:][path]filename1[,...] destination
```

Following are the description of the options which can be presented to the DEL command.

S.No.	Options & Description
1.	<b>[drive:][path]filename1</b> Specifies the location and name of the file or files you want to move
2.	<b>destination</b> Specifies the new location of the file. Destination can consist of a drive letter and colon, a directory name, or a combination. If you are moving only one file, you can also include a filename if you want to rename the file when you move it.
3.	<b>[drive:][path]dirname1</b> Specifies the directory you want to rename.
4.	<b>dirname2</b> Specifies the new name of the directory.
5.	<b>/Y</b> Suppresses prompting to confirm you want to overwrite an existing destination file.
6.	<b>/-Y</b> Causes prompting to confirm you want to overwrite an existing destination file.

Let's look at some examples of moving folders.

## Examples

```
move *.* C:\Example
```

The above command will move all files from the current directory to the folder C:\Example.

```
move *.txt C:\Example
```

The above command will move all files with the txt extension from the current directory to the folder C:\Example.

```
move C:\old\*.* C:\Example
```

The above command will move all files from the folder called 'old' in C drive to the folder C:\Example.

# Batch Script - Process

In this chapter, we will discuss the various processes involved in Batch Script.

## Viewing the List of Running Processes

In Batch Script, the TASKLIST command can be used to get the list of currently running processes within a system.

### Syntax

```
TASKLIST [/S system [/U username [/P [password]]]] [/M [module] | /SVC | /V] [/FI filter]  
[/FO format] [/NH]
```

Following are the description of the options which can be presented to the TASKLIST command.

### Examples

```
TASKLIST
```

The above command will get the list of all the processes running on your local system. Following is a snapshot of the output which is rendered when the above command is run as it is. As you can see from the following output, not only do you get the various processes running on your system, you also get the memory usage of each process.

Image Name	PID	Session Name	Session#	Mem Usage
System Idle Process	0	Services	0	4 K
System	4	Services	0	272 K
smss.exe	344	Services	0	1,040 K
csrss.exe	528	Services	0	3,892 K
csrss.exe	612	Console	1	41,788 K
wininit.exe	620	Services	0	3,528 K
winlogon.exe	648	Console	1	5,884 K
services.exe	712	Services	0	6,224 K
lsass.exe	720	Services	0	9,712 K
svchost.exe	788	Services	0	10,048 K
svchost.exe	832	Services	0	7,696 K
dwm.exe	916	Console	1	117,440 K
nvvsvc.exe	932	Services	0	6,692 K
nvxdsync.exe	968	Console	1	16,328 K
nvvsvc.exe	976	Console	1	12,756 K
svchost.exe	1012	Services	0	21,648 K
svchost.exe	236	Services	0	33,864 K
svchost.exe	480	Services	0	11,152 K
svchost.exe	1028	Services	0	11,104 K

svchost.exe	1048	Services	0	16,108 K
wlanext.exe	1220	Services	0	12,560 K
conhost.exe	1228	Services	0	2,588 K
svchost.exe	1276	Services	0	13,888 K
svchost.exe	1420	Services	0	13,488 K
spoolsv.exe	1556	Services	0	9,340 K

```
tasklist > process.txt
```

The above command takes the output displayed by tasklist and saves it to the process.txt file.

```
tasklist /fi "memusage gt 40000"
```

The above command will only fetch those processes whose memory is greater than 40MB. Following is a sample output that can be rendered.

Image Name	PID	Session Name	Session#	Mem Usage
dwm.exe	916	Console	1	127,912 K
explorer.exe	2904	Console	1	125,868 K
ServerManager.exe	1836	Console	1	59,796 K
WINWORD.EXE	2456	Console	1	144,504 K
chrome.exe	4892	Console	1	123,232 K
chrome.exe	4976	Console	1	69,412 K
chrome.exe	1724	Console	1	76,416 K
chrome.exe	3992	Console	1	56,156 K
chrome.exe	1168	Console	1	233,628 K
chrome.exe	816	Console	1	66,808 K

## Killing a Particular Process

Allows a user running Microsoft Windows XP professional, Windows 2003, or later to kill a task from a Windows command line by process id (PID) or image name. The command used for this purpose is the TASKKILL command.

### Syntax

```
TASKKILL [/S system [/U username [/P [password]]]] { [/FI filter]
[/PID processid | /IM imagename] } [/T] [/F]
```

Following are the description of the options which can be presented to the TASKKILL command.

### Examples

```
taskkill /f /im notepad.exe
```

The above command kills the open notepad task, if open.

```
taskkill /pid 9214
```

The above command kills a process which has a process of 9214.

## Starting a New Process

DOS scripting also has the availability to start a new process altogether. This is achieved by using the START command.

### Syntax

```
START "title" [/D path] [options] "command" [parameters]
```

Wherein

**title** – Text for the CMD window title bar (required.)

**path** – Starting directory.

**command** – The command, batch file or executable program to run.

**parameters** – The parameters passed to the command.

Following are the description of the options which can be presented to the START command.

### Examples

```
START "Test Batch Script" /Min test.bat
```

The above command will run the batch script test.bat in a new window. The windows will start in the minimized mode and also have the title of "Test Batch Script".

```
START "" "C:\Program Files\Microsoft Office\Winword.exe" "D:\test\TESTA.txt"
```

The above command will actually run Microsoft word in another process and then open the file TESTA.txt in MS Word.

## Batch Script - Aliases

Aliases means creating shortcuts or keywords for existing commands. Suppose if we wanted to execute the below command which is nothing but the directory listing command with the /w option to not show all of the necessary details in a directory listing.

```
Dir /w
```

Suppose if we were to create a shortcut to this command as follows.

```
dw = dir /w
```

When we want to execute the **dir /w** command, we can simply type in the word **dw**. The word 'dw' has now become an alias to the command Dir /w.

# Creating an Alias

Alias are managed by using the **doskey** command.

## Syntax

```
DOSKEY [options] [macroName=[text]]
```

Wherein

**macroName** – A short name for the macro.

**text** – The commands you want to recall.

Following are the description of the options which can be presented to the DOSKEY command.

S.No.	Options & Description
1.	<b>/REINSTALL</b> Installs a new copy of Doskey
2.	<b>/LISTSIZE = size</b> Sets size of command history buffer.
3.	<b>/MACROS</b> Displays all Doskey macros.
4.	<b>/MACROS:ALL</b> Displays all Doskey macros for all executables which have Doskey macros.
5.	<b>/MACROS:exename</b> Displays all Doskey macros for the given executable.
6.	<b>/HISTORY</b> Displays all commands stored in memory.
7.	<b>/INSERT</b> Specifies that new text you type is inserted in old text.
8.	<b>/OVERSTRIKE</b> Specifies that new text overwrites old text.

## Example

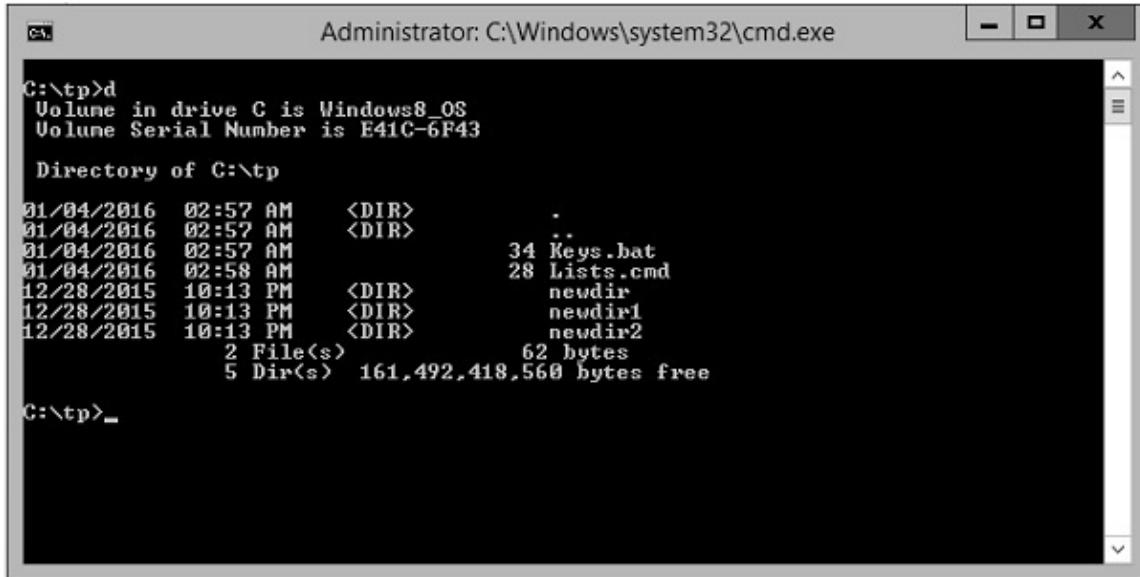
Create a new file called keys.bat and enter the following commands in the file. The below commands creates two aliases, one if for the cd command, which automatically goes to the directory called test. And the other is for the dir command.

```
@echo off  
doskey cd = cd/test  
doskey d = dir
```

Once you execute the command, you will able to run these aliases in the command prompt.

## Output

The following screenshot shows that after the above created batch file is executed, you can freely enter the 'd' command and it will give you the directory listing which means that your alias has been created.



A screenshot of a Windows Command Prompt window titled "Administrator: C:\Windows\system32\cmd.exe". The window displays the output of the "dir" command. The output shows the following directory listing:

```
C:\>tp>d  
Volume in drive C is Windows8_OS  
Volume Serial Number is E41C-6F43  
  
Directory of C:\tp  
01/04/2016  02:57 AM    <DIR>      .  
01/04/2016  02:57 AM    <DIR>      ..  
01/04/2016  02:57 AM                34 Keys.bat  
01/04/2016  02:58 AM                28 Lists.cmd  
12/28/2015  10:13 PM    <DIR>      newdir  
12/28/2015  10:13 PM    <DIR>      newdir1  
12/28/2015  10:13 PM    <DIR>      newdir2  
              2 File(s)           62 bytes  
              5 Dir(s)   161,492,418,560 bytes free  
  
C:\>tp>_
```

## Deleting an Alias

An alias or macro can be deleted by setting the value of the macro to NULL.

## Example

```
@echo off  
doskey cd = cd/test  
doskey d = dir  
d=
```

In the above example, we are first setting the macro d to d = dir. After which we are setting it to NULL. Because we have set the value of d to NULL, the macro d will deleted.

## Replacing an Alias

An alias or macro can be replaced by setting the value of the macro to the new desired

value.

## Example

```
@echo off  
doskey cd = cd/test  
doskey d = dir  
  
d = dir /w
```

In the above example, we are first setting the macro d to d = dir. After which we are setting it to dir /w. Since we have set the value of d to a new value, the alias 'd' will now take on the new value.

## Batch Script - Devices

Windows now has an improved library which can be used in Batch Script for working with devices attached to the system. This is known as the device console – DevCon.exe.

Windows driver developers and testers can use DevCon to verify that a driver is installed and configured correctly, including the proper INF files, driver stack, driver files, and driver package. You can also use the DevCon commands (enable, disable, install, start, stop, and continue) in scripts to test the driver. **DevCon** is a command-line tool that performs device management functions on local computers and remote computers.

Display driver and device info DevCon can display the following properties of drivers and devices on local computers, and remote computers (running Windows XP and earlier) –

Hardware IDs, compatible IDs, and device instance IDs. These identifiers are described in detail in device identification strings.

Device setup classes.

The devices in a device setup class.

INF files and device driver files.

Details of driver packages.

Hardware resources.

Device status.

Expected driver stack.

Third-party driver packages in the driver store.

Search for devices DevCon can search for installed and uninstalled devices on a local or remote computer by hardware ID, device instance ID, or device setup class.

Change device settings DevCon can change the status or configuration of Plug and Play (PnP) devices on the local computer in the following ways –

- Enable a device.
- Disable a device.
- Update drivers (interactive and non-interactive).
- Install a device (create a devnode and install software).
- Remove a device from the device tree and delete its device stack.
- Rescan for Plug and Play devices.
- Add, delete, and reorder the hardware IDs of root-enumerated devices.
- Change the upper and lower filter drivers for a device setup class.
- Add and delete third-party driver packages from the driver store.

DevCon (DevCon.exe) is included when you install the WDK, Visual Studio, and the Windows SDK for desktop apps. DevCon.exe kit is available in the following locations when installed.

```
%WindowsSdkDir%\tools\x64\devcon.exe  
%WindowsSdkDir%\tools\x86\devcon.exe  
%WindowsSdkDir%\tools\arm\devcon.exe
```

## Syntax

```
devcon [/m:\\computer] [/r] command [arguments]
```

wherein

**/m:\\computer** – Runs the command on the specified remote computer. The backslashes are required.

**/r** – Conditional reboot. Reboots the system after completing an operation only if a reboot is required to make a change effective.

**command** – Specifies a DevCon command.

To list and display information about devices on the computer, use the following commands –

- DevCon HwIDs
- DevCon Classes
- DevCon ListClass
- DevCon DriverFiles
- DevCon DriverNodes
- DevCon Resources
- DevCon Stack

DevCon Status

DevCon Dp\_enum

To search for information about devices on the computer, use the following commands –

DevCon Find

DevCon FindAll

To manipulate the device or change its configuration, use the following commands –

DevCon Enable

DevCon Disable

DevCon Update

DevCon UpdateNI

DevCon Install

DevCon Remove

DevCon Rescan

DevCon Restart

DevCon Reboot

DevCon SetHwID

DevCon ClassFilter

DevCon Dp\_add

DevCon Dp\_delete

## Examples

Following are some examples on how the DevCon command is used.

List all driver files

The following command uses the DevCon DriverFiles operation to list the file names of drivers that devices on the system use. The command uses the wildcard character (\*) to indicate all devices on the system. Because the output is extensive, the command uses the redirection character (>) to redirect the output to a reference file, driverfiles.txt.

devcon driverfiles \* > driverfiles.txt

The following command uses the DevCon status operation to find the status of all devices on the local computer. It then saves the status in the status.txt file for logging or later review. The command uses the wildcard character (\*) to represent all devices and the redirection character (>) to redirect the output to the status.txt file.

```
devcon status * > status.txt
```

The following command enables all printer devices on the computer by specifying the Printer setup class in a DevCon Enable command. The command includes the /r parameter, which reboots the system if it is necessary to make the enabling effective.

```
devcon /r enable = Printer
```

The following command uses the DevCon Install operation to install a keyboard device on the local computer. The command includes the full path to the INF file for the device (keyboard.inf) and a hardware ID (\*PNP030b).

```
devcon /r install c:\windows\inf\keyboard.inf *PNP030b
```

The following command will scan the computer for new devices.

```
devcon scan
```

The following command will rescan the computer for new devices.

```
devcon rescan
```

## Batch Script - Registry

The Registry is one of the key elements on a windows system. It contains a lot of information on various aspects of the operating system. Almost all applications installed on a windows system interact with the registry in some form or the other.

The Registry contains two basic elements: keys and values. **Registry keys** are container objects similar to folders. **Registry values** are non-container objects similar to files. Keys may contain values or further keys. Keys are referenced with a syntax similar to Windows' path names, using backslashes to indicate levels of hierarchy.

This chapter looks at various functions such as querying values, adding, deleting and editing values from the registry.

S.No	Types of Registry & Description
1	<b>Reading from the Registry</b> Reading from the registry is done via the REG QUERY command.
2	<b>Adding to the Registry</b> Adding to the registry is done via the REG ADD command.
3	<b>Deleting from the Registry</b> Deleting from the registry is done via the REG DEL command.
4	<b>Copying Registry Keys</b> Copying from the registry is done via the REG COPY command.

**Comparing Registry Keys**

Comparing registry keys is done via the REG COMPARE command.



# Batch Script - Reading from the Registry

Advertisements

Reading from the registry is done via the REG QUERY command. This command can be used to retrieve values of any key from within the registry.

## Syntax

```
REG QUERY [ROOT\]RegKey /v ValueName [/s]  
REG QUERY [ROOT\]RegKey /ve --This returns the (default) value
```

Where RegKey is the key which needs to be searched for in the registry.

## Example

```
@echo off R  
EG QUERY HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Windows\
```

The above command will query all the keys and their respective values under the registry key HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\Windows\

## Output

The output will display all the keys and values under the registry key.

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Windows\
```

This location in the registry has some key information about the windows system such as the System Directory location.

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Windows  
Directory REG_EXPAND_SZ %SystemRoot%  
SystemDirectory REG_EXPAND_SZ %SystemRoot%\system32  
NoInteractiveServices REG_DWORD 0x1  
CSDBuildNumber REG_DWORD 0x4000  
ShellErrorMode REG_DWORD 0x1  
ComponentizedBuild REG_DWORD 0x1  
CSDVersion REG_DWORD 0x0  
ErrorMode REG_DWORD 0x0  
CSDReleaseType REG_DWORD 0x0
```



# Batch Script - Adding to the Registry

Advertisements

Adding to the registry is done via the REG ADD command. Note that in order to add values to the registry you need to have sufficient privileges on the system to perform this operation.

## Syntax

The REG ADD command has the following variations. In the second variation, no name is specified for the key and it will add the name of "(Default)" for the key.

```
REG ADD [ROOT\]RegKey /v ValueName [/t DataType] [/S Separator] [/d Data] [/f]
REG ADD [ROOT\]RegKey /ve [/d Data] [/f]
```

Where

**ValueName** – The value, under the selected RegKey, to edit.

**/d Data** – The actual data to store as a "String", integer, etc.

**/f** – Force an update without prompting "Value exists, overwrite Y/N".

**/S Separator** – Character to use as the separator in REG\_MULTI\_SZ values. The default is "\0".

**/t DataType** – These are the data types defined as per the registry standards which can be –

REG\_SZ (default)

REG\_DWORD

REG\_EXPAND\_SZ

REG\_MULTI\_SZ

## Example

```
@echo off
REG ADD HKEY_CURRENT_USER\Console /v Test /d "Test Data"
REG QUERY HKEY_CURRENT_USER\Console /v Test
```

In the above example, the first part is to add a key into the registry under the location HKEY\_CURRENT\_USER\Console. This key will have a name of Test and the value assigned to the key will be Test Data which will be of the default string type.

The second command just displays what was added to the registry by using the REG QUERY command.

## Output

Following will be the output of the above program. The first line of the output shows that the 'Add' functionality was successful and the second output shows the inserted value into the registry.

```
The operation completed successfully.  
HKEY_CURRENT_USER\Console  
    Test REG_SZ Test Data
```

# Batch Script - Deleting from the Registry

Advertisements

Deleting from the registry is done via the REG DEL command. Note that in order to delete values from the registry you need to have sufficient privileges on the system to perform this operation.

## Syntax

The REG DELETE command has the following variations. In the second variation, the default value will be removed and in the last variation all the values under the specified key will be removed.

```
REG DELETE [ROOT\]RegKey /v ValueName [/f]
REG DELETE [ROOT\]RegKey /ve [/f]
REG DELETE [ROOT\]RegKey /va [/f]
```

Where

**ValueName** – The value, under the selected RegKey, to edit.

**/f** – Force an update without prompting "Value exists, overwrite Y/N".

## Example

```
@echo off
REG DELETE HKEY_CURRENT_USER\Console /v Test /f
REG QUERY HKEY_CURRENT_USER\Console /v Test
```

In the above example, the first part is to delete a key into the registry under the location HKEY\_CURRENT\_USER\Console. This key has the name of Test. The second command just displays what was deleted to the registry by using the REG QUERY command. From this command, we should expect an error, just to ensure that our key was in fact deleted.

## Output

Following will be the output of the above program. The first line of the output shows that the 'Delete' functionality was successful and the second output shows an error which was expected to confirm that indeed our key was deleted from the registry.



# Batch Script - Copying Registry Keys

Advertisements

Copying from the registry is done via the REG COPY command. Note that in order to copy values from the registry, you need to have sufficient privileges on the system to perform this operation on both the source location and the destination location.

## Syntax

```
REG COPY [\SourceMachine\[ROOT\]RegKey [\DestMachine\[ROOT\]RegKey
```

## Example

```
@echo off
REG COPY HKEY_CURRENT_USER\Console HKEY_CURRENT_USER\Console\Test
REG QUERY HKEY_CURRENT_USER\Console\Test
```

In the above example, the first part is to copy the contents from the location HKEY\_CURRENT\_USER\Console into the location HKEY\_CURRENT\_USER\Console\Test on the same machine. The second command is used to query the new location to check if all the values were copied properly.

## Output

Following is the output of the above program. The first line of the output shows that the 'Copy' functionality was successful and the second output shows the values in our copied location.

```
The operation completed successfully.

HKEY_CURRENT_USER\Console\Test
    HistoryNoDup REG_DWORD 0x0
    FullScreen REG_DWORD 0x0
    ScrollScale REG_DWORD 0x1
    ExtendedEditKeyCustom REG_DWORD 0x0
    CursorSize REG_DWORD 0x19
    FontFamily REG_DWORD 0x0
    ScreenColors REG_DWORD 0x7
    TrimLeadingZeros REG_DWORD 0x0
```



# Batch Script - Comparing Registry Keys

Advertisements

Comparing registry keys is done via the REG COMPARE command.

## Syntax

```
REG COMPARE [ROOT\]RegKey [ROOT\]RegKey [/v ValueName] [Output] [/s]
REG COMPARE [ROOT\]RegKey [ROOT\]RegKey [/ve] [Output] [/s]
```

Wherein **Output** – /od (only differences) /os (only matches) /oa (all) /on (no output).

## Example

```
@echo off
REG COMPARE HKEY_CURRENT_USER\Console HKEY_CURRENT_USER\Console\Test
```

The above program will compare all of the values between the registry keys HKEY\_CURRENT\_USER\Console & HKEY\_CURRENT\_USER\Console\Test.

## Output

```
Result Compared: Identical
The operation completed successfully.
```

If there is a difference between the values in either registry key, it will be shown in the output as shown in the following result. The following output shows that the value 'EnableColorSelection' is extra I the registry key 'HKEY\_CURRENT\_USER\Console'.

```
< Value: HKEY_CURRENT_USER\Console EnableColorSelection REG_DWORD 0x0
Result Compared: Different
The operation completed successfully.
```

# Batch Script - Network

Batch script has the facility to work with network settings. The NET command is used to update, fix, or view the network or network settings. This chapter looks at the different options available for the net command.

S.No	NET Commands & Description
	<b>NET ACCOUNTS</b>
1	View the current password & logon restrictions for the computer.
	<b>NET CONFIG</b>
2	Displays your current server or workgroup settings.
	<b>NET COMPUTER</b>
3	Adds or removes a computer attached to the windows domain controller.
	<b>NET USER</b>
4	This command can be used for the following View the details of a particular user account.
	<b>NET STOP/START</b>
5	This command is used to stop and start a particular service.
	<b>NET STATISTICS</b>
6	Display network statistics of the workstation or server.
	<b>NET USE</b>
7	Connects or disconnects your computer from a shared resource or displays information about your connections.



# Batch Script - NET ACCOUNTS

Advertisements

View the current password & logon restrictions for the computer.

## Syntax

```
NET ACCOUNT [/FORCELOGOFF:{minutes | NO}] [/MINPWLEN:length]
[/MAXPWAGE:{days | UNLIMITED}] [/MINPWAGE:days]
[/UNIQUEPW:number] [/DOMAIN]
```

Wherein

**FORCELOGOFF** – Force the log-off of the current user within a defined time period.

**MINPWLEN** – This is the minimum password length setting to provide for the user.

**MAXPWAGE** – This is the maximum password age setting to provide for the user.

**MINPWAGE** – This is the minimum password age setting to provide for the user.

## Example

```
NET ACCOUNT
```

## Output

```
Force user logoff how long after time expires?: Never
Minimum password age (days): 0
Maximum password age (days): 42
Minimum password length: 0
Length of password history maintained: None
Lockout threshold: Never
Lockout duration (minutes): 30
Lockout observation window (minutes): 30
Computer role: SERVER
The command completed successfully.
```



# Batch Script - NET CONFIG

Advertisements

Displays your current server or workgroup settings.

## Syntax

```
NET CONFIG
```

## Example

```
NET CONFIG
```

## Output

```
The following running services can be controlled:
```

```
Server
```

```
Workstation
```

```
The command completed successfully.
```



# Batch Script - NET COMPUTER

Advertisements

Adds or removes a computer attached to the windows domain controller.

## Syntax

```
NET COMPUTER \\computername {/ADD | /DEL}
```

## Example

```
NET COMPUTER \\dxbtest /ADD
```

## Output

The above command will add the machine with the name dxbtest to the domain in which the windows domain controller exists.



# Batch Script - NET USER

Advertisements

This command can be used for the following –

- View the details of a particular user account.
- Add a user account.
- Delete a user's account.
- Modify a user's account.

## Syntax

```
Net user [username [password | *] [options]] [/DOMAIN]
username {password | *} /ADD [options] [/DOMAIN]
username [/DELETE] [/DOMAIN]
```

## Example

```
NET USER
```

The above command shows all the accounts defined on a system. Following is the output of the above command.

```
User accounts for \\WIN-50GP30FG075
-----
Administrator      atlbitbucket      Guest
The command completed successfully.
```

```
net user Guest
```

The above command shows the details of Guest account defined on a system. Following is the output of the above command.

## Output

User name	Guest
Full Name	
Comment	Built-in account for guest access to the computer/domain

User's comment

Country/region code 000 (System Default)

Account active No

Account expires Never

Password last set 1/4/2016 9:34:25 AM

Password expires Never

Password changeable 1/4/2016 9:34:25 AM

Password required No

User may change password No

Workstations allowed All

Logon script

User profile

Home directory

Last logon Never

Logon hours allowed All

Local Group Memberships \*Guests

Global Group memberships \*None

The command completed successfully.



# Batch Script - NET STOP/START

Advertisements

This command is used to stop and start a particular service.

## Syntax

```
Net stop/start [servicename]
```

## Example

```
NET STOP Spooler
```

The above command is used to stop the printer spooler service. Following is the output of the above command.

```
The Print Spooler service is stopping.  
The Print Spooler service was stopped successfully.  
NET START Spooler
```

The above command is used to start the printer spooler service. Following is the output of the above command.

```
The Print Spooler service is starting.  
The Print Spooler service was started successfully.
```



# Batch Script - NET STATISTICS

Advertisements

Display network statistics of the workstation or server.

## Syntax

```
Net statistics [SERVER/WORKSTATION]
```

## Example

```
Net statistics Server
```

## Output

```
Server Statistics for \\WIN-50GP30FG075
```

```
Statistics since 1/3/2016 9:16:28 PM
```

```
Sessions accepted 0
```

```
Sessions timed-out 0
```

```
Sessions errored-out 0
```

```
Kilobytes sent 0
```

```
Kilobytes received 0
```

```
Mean response time (msec) 0
```

```
System errors 0
```

```
Permission violations 0
```

```
Password violations 0
```

```
Files accessed 0
```

```
Communication devices accessed 0
```

```
Print jobs spooled 0
```



# Batch Script - NET USE

Advertisements

Connects or disconnects your computer from a shared resource or displays information about your connections.

## Syntax

```
NET USE [devicename | *] [\computername\sharename[\volume] [password | *]]  
[/USER:[domainname\]username]  
[/USER:[dotted domain name\]username]  
[/USER:[username@dotted domain name]  
[/SMARTCARD]  
[/SAVECRED] [[/DELETE] | [/PERSISTENT:{YES | NO}]]
```

where

**\computername\sharename** – This is the name of the share which needs to be connected to.

**/USER** – This needs to be specified to ensure that the right credentials are specified when connecting to the network share.

## Example

```
net use z: \\computer\test
```

The above command will connect to the share name \\computer\test and assign the Z: drive name to it.

# Batch Script - Printing

Printing can also be controlled from within Batch Script via the NET PRINT command.

## Syntax

```
PRINT [/D:device] [[drive:][path]filename[...]]
```

Where /D:device - Specifies a print device.

## Example

```
print c:\example.txt /c /d:lpt1
```

The above command will print the example.txt file to the parallel port lpt1.

## Command Line Printer Control

As of Windows 2000, many, but not all, printer settings can be configured from Windows's command line using PRINTUI.DLL and RUNDLL32.EXE

### Syntax

```
RUNDLL32.EXE PRINTUI.DLL,PrintUIEntry [ options ] [ @commandfile ]
```

Where some of the options available are the following –

- /dl** – Delete local printer.
- /dn** – Delete network printer connection.
- /dd** – Delete printer driver.
- /e** – Display printing preferences.
- /f[file]** – Either inf file or output file.
- /F[file]** – Location of an INF file that the INF file specified with /f may depend on.
- /ia** – Install printer driver using inf file.
- /id** – Install printer driver using add printer driver wizard.
- /if** – Install printer using inf file.
- /ii** – Install printer using add printer wizard with an inf file.
- /il** – Install printer using add printer wizard.
- /in** – Add network printer connection.
- /ip** – Install printer using network printer installation wizard.
- /k** – Print test page to specified printer, cannot be combined with command when installing a printer.
- /l[path]** – Printer driver source path.
- /m[model]** – Printer driver model name.
- /n[name]** – Printer name.
- /o** – Display printer queue view.
- /p** – Display printer properties.
- /Ss** – Store printer settings into a file.

**/Sr** – Restore printer settings from a file.

**/y** – Set printer as the default.

**/Xg** – Get printer settings.

**/Xs** – Set printer settings.

## Testing if a Printer Exists

There can be cases wherein you might be connected to a network printer instead of a local printer. In such cases, it is always beneficial to check if a printer exists in the first place before printing.

The existence of a printer can be evaluated with the help of the RUNDLL32.EXE PRINTUI.DLL which is used to control most of the printer settings.

### Example

```
SET PrinterName = Test Printer
SET file=%TEMP%\Prt.txt
RUNDLL32.EXE PRINTUI.DLL,PrintUIEntry /Xg /n "%PrinterName%" /f "%file%" /q

IF EXIST "%file%" (
    ECHO %PrinterName% printer exists
) ELSE (
    ECHO %PrinterName% printer does NOT exists
)
```

The above command will do the following –

It will first set the printer name and set a file name which will hold the settings of the printer.

The RUNDLL32.EXE PRINTUI.DLL commands will be used to check if the printer actually exists by sending the configuration settings of the file to the file Prt.txt

## Batch Script - Debugging

Very often than not you can run into problems when running batch files and most often than not you would need to debug your batch files in some way or the other to determine the issue with the batch file itself. Following are some of the techniques that can help in debugging Batch Script files.

## Error Messages

To discover the source of the message, follow these steps –

**Step 1** – REM out the @ECHO OFF line, i.e. REM @ECHO OFF or :: @ECHO OFF.

**Step 2** – Run the batch file with the required command line parameters, redirecting all output to a log file for later comparison.

```
test.bat > batch.log 2>&1
```

**Step 3** – Search the file batch.log for the error messages

**Step 4** – Check the previous line for any unexpected or invalid command, command line switch(es) or value(s); pay special attention to the values of any environment variables used in the command.

**Step 5** – Correct the error and repeat this process until all error messages have disappeared.

## Complex Command Lines

Another common source of errors are incorrectly redirected commands, like for example "nested" FIND or FINDSTR commands with incorrect search strings, sometimes within a FOR /F loop.

To check the validity of these complex commands, follow these steps –

**Step 1** – Insert "command check lines" just before a line which uses the complex command set.

Following is an example wherein the ECHO command is inserted to mark where the output of the first TYPE command ends and the next one starts.

```
TYPE %Temp%\apipaorg.reg
ECHO.=====
| FIND
"[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\TCPIP\Parameters\Interfaces\"
```

**Step 2** – Follow the procedure to find error message sources described above.

**Step 3** – Pay special attention to the output of the "simplified" command lines: Is the output of the expected format? Is the "token" value or position as expected?

## Subroutines

Subroutines generating error messages pose an extra "challenge" in finding the cause of the error, as they may be called multiple times in the same batch file.

To help find out what causes the incorrect call to the subroutine, follow these steps –

**Step 1** – Add and reset a counter variable at the beginning of the script –

```
SET Counter = 0
```

**Step 2** – Increment the counter each time the subroutine is called, by inserting the following line at the beginning of the subroutine

```
SET /A Counter+=1
```

**Step 3** – Insert another line right after the counter increment, containing only the SET

command; this will list all environment variables and their values.

**Step 4** – Follow the procedure to find error message sources described above.

## Windows Versions

If you intend to distribute your batch files to other computers that may or may not run the same Windows version, you will need to test your batch files in as many Windows versions as possible.

The following example shows how to check for various operating system versions to check the relevant windows versions.

```
@ECHO OFF
:: Check for Windows NT 4 and later

IF NOT "%OS%"=="Windows_NT" GOTO DontRun
:: Check for Windows NT 4
VER | FIND "Windows NT" >NUL && GOTO DontRun
:: Check for Windows 2000
VER | FIND "Windows 2000" >NUL && GOTO DontRun
:: Place actual code here . . .
:: End of actual code . . .
EXIT

:DontRun
ECHO Sorry, this batch file was written for Windows XP and later versions only
```

## Batch Script - Logging

Logging in is possible in Batch Script by using the redirection command.

## Syntax

```
test.bat > testlog.txt 2> testerrors.txt
```

## Example

Create a file called test.bat and enter the following command in the file.

```
net statistics /Server
```

The above command has an error because the option to the net statistics command is given in the wrong way.

## Output

If the command with the above test.bat file is run as

```
test.bat > testlog.txt 2> testerrors.txt
```

And you open the file testerrors.txt, you will see the following error.

The option /SERVER is unknown.

The syntax of this command is –

```
NET STATISTICS  
[WORKSTATION | SERVER]
```

More help is available by typing NET HELPMSG 3506.

If you open the file called testlog.txt, it will show you a log of what commands were executed.

```
C:\tp>net statistics /Server
```

Para obter mais informações sobre um comando específico,  
digite HELP nome\_do\_comando

ASSOC	Exibe ou modifica associações de extensões de arquivo.
ATTRIB	Exibe ou altera atributos de arquivos.
BREAK	Define ou limpa a verificação estendida CTRL+C.
BCDEDIT	Define propriedades no banco de dados de inicialização para controlar o carregamento da inicialização.
CACLS	Exibe ou modifica listas de controle de acesso de arquivos.
CALL	Chama um programa em lotes por meio de outro.
CD	Exibe o nome do diretório atual ou faz alterações nele.
CHCP	Exibe ou define o número da página de código ativa.
CHDIR	Exibe o nome do diretório atual ou faz alterações nele.
CHKDSK	Verifica um disco e exibe um relatório de status.
CHKNTFS	Exibe ou modifica a verificação do disco na inicialização.
CLS	Limpa a tela.
CMD	Inicia uma nova instância do interpretador de comandos do Windows.
COLOR	Define as cores padrão do primeiro plano e da tela de fundo do console.
COMP	Compara o conteúdo de dois arquivos ou grupos de arquivos.
COMPACT	Exibe ou altera a compactação de arquivos em partições NTFS.
CONVERT	Converte volumes FAT em NTFS. Não é possível converter a unidade atual.
COPY	Copia um ou mais arquivos para outro local.
DATE	Exibe ou ajusta a data.
DEL	Exclui um ou mais arquivos.
DIR	Exibe uma lista de arquivos e subdiretórios em um diretório.
DISKCOMP	Compara o conteúdo de dois disquetes.
DISKCOPY	Copia o conteúdo de um disquete para outro.
DISKPART	Exibe ou configura propriedades de partição de disco.
DOSKEY	Edita linhas de comando, volta a chamar comandos do Windows e cria macros.
DRIVERQUERY	Exibe status e propriedades do driver de dispositivo atual.
ECHO	Exibe mensagens ou ativa/desativa o eco de comandos.
ENDLOCAL	Encerra a localização de alterações de ambiente em um arquivo em lotes.
ERASE	Exclui um ou mais arquivos.
EXIT	Sai do programa CMD.EXE (interpretador de comandos).
FC	Compara dois arquivos ou grupos de arquivos e exibe as diferenças entre eles.
FIND	Procura uma cadeia de caracteres de texto em um ou mais arquivos.
FINDSTR	Procura cadeias de caracteres em arquivos.
FOR	Executa um determinado comando para cada arquivo em um grupo de arquivos.
FORMAT	Formata um disco para ser usado com o Windows.
FSUTIL	Exibe ou configura as propriedades do sistema de arquivos.
FTYPE	Exibe ou modifica os tipos de arquivo utilizados em associações de extensões de arquivo.
GOTO	Direciona o interpretador de comandos do Windows para uma linha identificada em um programa em lotes.
GPRESULT	Exibe informações de política de grupo para o computador ou usuário.
GRAFTABL	Permite que o Windows exiba um conjunto de caracteres estendidos em modo gráfico.
HELP	Fornece informações de ajuda sobre comandos do Windows.
ICACLS	Exibir, modificar, fazer backup ou restaurar ACLs de arquivos e diretórios.
IF	Realiza processamento condicional em arquivos em lotes.
LABEL	Cria, altera ou exclui o rótulo de volume de um disco.
MD	Cria um diretório.
MKDIR	Cria um diretório.
MKLINK	Cria Vínculos Simbólicos e Links Físicos
MODE	Configura um dispositivo do sistema.
MORE	Exibe a saída fazendo pausa a cada tela.

MOVE	Move um ou mais arquivos de um diretório para outro diretório.
OPENFILES	Exibe arquivos abertos por usuários remotos para um compartilhamento de arquivo.
PATH	Exibe ou define um caminho de pesquisa para arquivos executáveis.
PAUSE	Suspende o processamento de um arquivo em lotes e exibe uma mensagem.
POPD	Restaura o valor anterior do diretório atual salvo por PUSHD.
PRINT	Imprime um arquivo de texto.
PROMPT	Altera o prompt de comando do Windows.
PUSHD	Salva o diretório atual e o altera em seguida.
RD	Remove um diretório.
RECOVER	Recupera informações legíveis de um disco defeituoso ou danificado.
REM	Grava comentários (observações) em arquivos em lotes ou no CONFIG.SYS.
REN	Altera o nome de um ou mais arquivos.
RENAME	Altera o nome de um ou mais arquivos.
REPLACE	Substitui arquivos.
RMDIR	Remove um diretório.
ROBOCOPY	Utilitário avançado para copiar arquivos e árvores de diretório
SET	Exibe, define ou remove variáveis de ambiente do Windows.
SETLOCAL	Inicia a localização de alterações de ambiente em um arquivo em lotes.
SC	Exibe ou configura serviços (processos em segundo plano).
SCHTASKS	Agenda a execução de comandos e programas em um computador.
SHIFT	Altera a posição dos parâmetros substituíveis em um arquivo em lotes.
SHUTDOWN	Permite o desligamento local ou remoto adequado do computador.
SORT	Classifica a entrada.
START	Inicia uma janela separada para executar um programa ou comando especificado.
SUBST	Associa um caminho a uma letra de unidade.
SYSTEMINFO	Exibe a configuração e propriedades específicas da máquina.
TASKLIST	Exibe todas as tarefas em execução no momento, inclusive serviços.
TASKKILL	Termina ou interrompe um processo ou aplicativo em execução.
TIME	Exibe ou ajusta a hora do sistema.
TITLE	Define o título da janela para uma sessão do CMD.EXE.
TREE	Exibe graficamente a estrutura de diretórios de uma unidade ou caminho.
TYPE	Exibe o conteúdo de um arquivo de texto.
VER	Exibe a versão do Windows.
VERIFY	Faz com que o Windows verifique se os arquivos estão gravados corretamente em um disco.
VOL	Exibe o rótulo de volume e o número de série de um disco.
XCOPY	Copia arquivos e árvores de diretórios.
WMIC	Exibe informações WMI em um comando interativo do shell.

Para obter mais informações sobre ferramentas, consulte a referência a linhas de comando na ajuda online.

Para obter mais informações sobre um comando específico,  
digite HELP nome\_do\_comando

ASSOC	Exibe ou modifica associações de extensões de arquivo.
ATTRIB	Exibe ou altera atributos de arquivos.
BREAK	Define ou limpa a verificação estendida CTRL+C.
BCDEDIT	Define propriedades no banco de dados de inicialização
para	controlar o carregamento da inicialização.
CACLS	Exibe ou modifica listas de controle de acesso de
arquivos.	
CALL	Chama um programa em lotes por meio de outro.
CD	Exibe o nome do diretório atual ou faz alterações nele.
CHCP	Exibe ou define o número da página de código ativa.
CHDIR	Exibe o nome do diretório atual ou faz alterações nele.
CHKDSK	Verifica um disco e exibe um relatório de status.
CHKNTFS	Exibe ou modifica a verificação do disco na inicialização.
CLS	Limpa a tela.
CMD	Inicia uma nova instância do interpretador de comandos do Windows.
COLOR	Define as cores padrão do primeiro plano e da tela de
fundo	do console.
COMP	Compara o conteúdo de dois arquivos ou grupos de arquivos.
COMPACT	Exibe ou altera a compactação de arquivos em partições
NTFS.	
CONVERT	Converte volumes FAT em NTFS. Não é possível converter a unidade atual.
COPY	Copia um ou mais arquivos para outro local.
DATE	Exibe ou ajusta a data.
DEL	Exclui um ou mais arquivos.
DIR	Exibe uma lista de arquivos e subdiretórios em um
diretório.	
DISKCOMP	Compara o conteúdo de dois disquetes.
DISKCOPY	Copia o conteúdo de um disquete para outro.
DISKPART	Exibe ou configura propriedades de partição de disco.
DOSKEY	Edita linhas de comando, volta a chamar comandos do
Windows e	cria macros.
DRIVERQUERY	Exibe status e propriedades do driver de dispositivo
atual.	
ECHO	Exibe mensagens ou ativa/desativa o eco de comandos.
ENDLOCAL	Encerra a localização de alterações de ambiente em um
arquivo	em lotes.
ERASE	Exclui um ou mais arquivos.
EXIT	Sai do programa CMD.EXE (interpretador de comandos).
FC	Compara dois arquivos ou grupos de arquivos e exibe as diferenças entre eles.
FIND	Procura uma cadeia de caracteres de texto em um ou mais arquivos.
FINDSTR	Procura cadeias de caracteres em arquivos.
FOR	Executa um determinado comando para cada arquivo em um
grupo	de arquivos.
FORMAT	Formata um disco para ser usado com o Windows.
FSUTIL	Exibe ou configura as propriedades do sistema de arquivos.
FTYPE	Exibe ou modifica os tipos de arquivo utilizados em associações de extensões de arquivo.
GOTO	Direciona o interpretador de comandos do Windows para uma linha identificada em um programa em lotes.

GPRESULT	Exibe informações de política de grupo para o computador ou usuário.
GRAFTABL	Permite que o Windows exiba um conjunto de caracteres estendidos em modo gráfico.
HELP	Fornece informações de ajuda sobre comandos do Windows.
ICACLS	Exibir, modificar, fazer backup ou restaurar ACLs de arquivos e diretórios.
IF	Realiza processamento condicional em arquivos em lotes.
LABEL	Cria, altera ou exclui o rótulo de volume de um disco.
MD	Cria um diretório.
MKDIR	Cria um diretório.
MKLINK	Cria Vínculos Simbólicos e Links Físicos
MODE	Configura um dispositivo do sistema.
MORE	Exibe a saída fazendo pausa a cada tela.
MOVE	Move um ou mais arquivos de um diretório para outro diretório.
OPENFILES	Exibe arquivos abertos por usuários remotos para um compartilhamento de arquivo.
PATH	Exibe ou define um caminho de pesquisa para arquivos executáveis.
PAUSE	Suspende o processamento de um arquivo em lotes e exibe uma mensagem.
POPD	Restaura o valor anterior do diretório atual salvo por
PUSHD.	
PRINT	Imprime um arquivo de texto.
PROMPT	Altera o prompt de comando do Windows.
PUSHD	Salva o diretório atual e o altera em seguida.
RD	Remove um diretório.
RECOVER	Recupera informações legíveis de um disco defeituoso ou danificado.
REM	Grava comentários (observações) em arquivos em lotes ou no CONFIG.SYS.
REN	Altera o nome de um ou mais arquivos.
RENAME	Altera o nome de um ou mais arquivos.
REPLACE	Substitui arquivos.
RMDIR	Remove um diretório.
ROBOCOPY	Utilitário avançado para copiar arquivos e árvores de diretório
SET	Exibe, define ou remove variáveis de ambiente do Windows.
SETLOCAL	Inicia a localização de alterações de ambiente em um arquivo
	em lotes.
SC	Exibe ou configura serviços (processos em segundo plano).
SCHTASKS	Agenda a execução de comandos e programas em um computador.
SHIFT	Altera a posição dos parâmetros substituíveis em um arquivo em.
	lotes.
SHUTDOWN	Permite o desligamento local ou remoto adequado do computador.
SORT	Classifica a entrada.
START	Inicia uma janela separada para executar um programa ou comando especificado.
SUBST	Associa um caminho a uma letra de unidade.
SYSTEMINFO	Exibe a configuração e propriedades específicas da máquina.
TASKLIST	Exibe todas as tarefas em execução no momento, inclusive serviços.

TASKKILL	Termina ou interrompe um processo ou aplicativo em execução.
TIME	Exibe ou ajusta a hora do sistema.
TITLE	Define o título da janela para uma sessão do CMD.EXE.
TREE	Exibe graficamente a estrutura de diretórios de uma unidade
	ou caminho.
TYPE	Exibe o conteúdo de um arquivo de texto.
VER	Exibe a versão do Windows.
VERIFY	Faz com que o Windows verifique se os arquivos estão gravados
	corretamente em um disco.
VOL	Exibe o rótulo de volume e o número de série de um disco.
XCOPY	Copia arquivos e árvores de diretórios.
WMIC	Exibe informações WMI em um comando interativo do shell.

Para obter mais informações sobre ferramentas, consulte a referência a linhas de comando na ajuda online.

Exibe ou modifica associações com extensões de arquivos

ASSOC [.ext[=[Tipo\_de\_arquivo]]]

- .ext Especifica a extensão de arquivo com a qual associar
- o tipo de arquivo
- Tipo\_de\_arquivo Especifica o tipo de arquivo a ser associado com a extensão de arquivo

Digite ASSOC sem parâmetros para exibir as atuais associações de arquivos.

Se ASSOC for chamado com apenas uma extensão de arquivo, exibirá a atual associação de arquivos para essa extensão de arquivo. Se nada for especificado para o tipo de arquivo, o comando excluirá a associação para a extensão de arquivo.

Exibe ou altera atributos de arquivo.

ATTRIB [+R | -R] [+A | -A] [+S | -S] [+H | -H] [+I | -I]  
[unidade:] [caminho] [arquivo] [/S [/D] [/L]]

+ Define um atributo.

- Limpa um atributo.

R Atributo de arquivo somente leitura.

A Atributo de arquivo morto.

S Atributo de arquivo do sistema.

H Atributo de arquivo oculto.

I Atributo de arquivo sem conteúdo indexado.

X Sem atributo de arquivo de limpeza.

V Atributo de integridade.

[unidade:] [caminho] [arquivo]

Especifica um ou mais arquivos para processamento de atributos.

/S Processa os arquivos correspondentes na pasta atual  
e em todas as subpastas.

/D Inclui pastas no processamento.

/L Trabalha nos atributos do Link Simbólico versus  
o destino do Link Simbólico

## BCDEDIT - Editor de Repositório de Dados de Configuração da Inicialização

A ferramenta de linha de comando Bcdedit.exe modifica o repositório de dados

de configuração da inicialização. Este repositório contém parâmetros de configuração da inicialização e controla o modo como o sistema operacional

é inicializado. Esses parâmetros estavam anteriormente no arquivo Boot.ini

(nos sistemas operacionais baseados em BIOS) ou nas entradas de RAM não voláteis (nos sistemas operacionais baseados em EFI). Você pode usar o Bcdedit.exe para adicionar, excluir, editar e anexar entradas no repositório de dados de configuração da inicialização.

Para obter informações detalhadas sobre comandos e opções, digite bcdedit.exe /? <comando>. Por exemplo, para exibir informações detalhadas sobre o comando /createstore, digite:

```
bcdedit.exe /? /createstore
```

Para obter uma lista em ordem alfabética dos tópicos neste arquivo de ajuda, execute "bcdedit /? TOPICS".

Comandos que operam em um repositório

```
=====
```

/createstore Cria um repositório de dados de configuração da inicialização novo e vazio.

/export Exporta o conteúdo do repositório do sistema para um arquivo.

Este arquivo pode ser usado depois para restaurar o estado do repositório do sistema.

/import Restaura o estado do repositório do sistema usando um arquivo

de backup criado com o comando /export.

/sysstore Define o dispositivo de repositório do sistema (afeta somente sistemas EFI, não persiste depois de reinicializações, e só

é usado em casos em que o dispositivo de repositório do sistema é ambíguo).

Comandos que operam nas entradas em um repositório

```
=====
```

/copy Faz cópias de entradas no repositório.

/create Cria novas entradas no repositório.

/delete Exclui entradas do repositório.

/mirror Cria espelhos de entradas do repositório.

Execute bcdedit /? ID para obter informações sobre identificadores usados por esses comandos.

Comandos que operam nas opções de entrada

```
=====
```

/deletevalue Exclui opções de entrada do repositório.

/set Define valores de opção de entrada no repositório.

Execute bcdedit /? TYPES para obter uma lista de tipos de dados usados por  
esses comandos.

Execute bcdedit /? FORMATS para obter uma lista de formatos de dados válidos.

Comandos que controlam a saída

```
=====
/enum           Lista entradas no repositório.
/v              Opção de linha de comando que exibe identificadores
                de entrada completos, em vez de usar nomes para
                identificadores conhecidos. Use /v sozinho como um
comando          comando
                para exibir identificadores de entrada completos para o
tipo            tipo
                ACTIVE.
```

Executar "bcdedit" sozinho é equivalente a executar "bcdedit /enum ACTIVE".

Comandos que controlam o gerenciador de inicialização

```
=====
/bootsequence   Define a sequência de inicialização única para o
gerenciador
                de inicialização.
/default        Define a entrada padrão que o gerenciador de
inicialização
                usará.
/displayorder  Define a ordem na qual o gerenciador de inicialização
exibe o
                menu de inicialização múltipla.
/timeout        Define o valor do tempo limite do gerenciador de
inicialização.
/toolsdisplayorder Define a ordem na qual o gerenciador de inicialização
exibe o menu de ferramentas.
```

Comandos que controlam Serviços de Gerenciamento de Emergência  
para um aplicativo de inicialização

```
=====
/bootems       Habilita ou desabilita Serviços de Gerenciamento
                de Emergência para um aplicativo de inicialização.
/ems           Habilita ou desabilita Serviços de Gerenciamento
                de Emergência para uma entrada de sistema operacional.
/emssettings  Define os parâmetros globais dos Serviços de
Gerenciamento
                de Emergência.
```

Comando que controla a depuração

```
=====
/bootdebug     Habilita ou desabilita a depuração da inicialização para
um
                aplicativo de inicialização.
/dbgsettings  Define os parâmetros globais do depurador.
/debug         Habilita ou desabilita a depuração de kernel para uma
entrada
                do sistema operacional.
/hypervisorsettings Define os parâmetros do hipervisor.
```

Ativa ou desativa a verificação estendida de CTRL+C em sistemas DOS

Existe somente para manter a compatibilidade com sistemas DOS. Não tem efeito sob o Windows.

Se as extensões de comando estiverem ativadas e sendo executadas na plataforma Windows, o comando BREAK incluirá um ponto de interrupção codificado se estiver sendo depurado por um depurador.

OBS.: Cacls foi preterido; use Icacls.

Exibe/modifica listas de controle de acesso (ACLs) de arquivos.

```
CACLS nome_arquivo [/T] [/M] [/L] [/S[:SDDL]] [/E] [/C] [/G
usu_rio:perm]
    [/R usu_rio [...] ] [/P usu_rio:perm [...] ] [/D usu_rio [...] ]
    nome_arquivo Exibe ACLs.
    /T            Altera ACLs de arqs. especificados no dir. e nos
subdirs.
    /L            Trabalha no pr prio V nculo Simb lico, n o no destino.
    /M            Altera ACLs de volumes montados em diret rio.
    /S            Exibe a cadeia de caracteres SDDL para a DACL.
    /S:SDDL      Substitui ACLs pelas especificadas na cadeia de
caractere SDDL (n o , v lido com /E, /G, /R, /P ou
/D).
    /E            Edita a ACL, sem substitu -la.
    /C            Continua nos erros de acesso negado.
    /G usu_rio:perm Concede direitos de acesso ao usu_rio especificado.
        Perm pode ser: R Ler
                        W Gravar
                        C Alterar (gravar)
                        F Controle total
    /R usu_rio     Revoga direitos de acesso do usu_rio especificado
                    (v lido s  com /E).
    /P usu_rio:perm Substitui direitos de acesso do usu_rio
especificado.
        Perm pode ser: N Nenhum
                        R Ler
                        W Gravar
                        C Alterar (gravar)
                        F Controle total
    /D usu_rio     Nega acesso ao usu_rio especificado.
Caracteres curinga podem ser usados para especificar mais de um arquivo
em
um comando. Voc  pode especificar mais de um usu_rio em um comando.
Abrevia es:
    CI - Heran a por Cont iner.
        A ACE ser herdada por diret rios.
    OI - Heran a por Objeto.
        A ACE ser herdada por arquivos.
    IO - Somente Heran a.
        A ACE n o se aplica ao arquivo/diret rio atual.
    ID - Herdado.
        A ACE foi herdada da ACL do diret rio pai.
```

Chama um programa em lotes de outro.

CALL [unidade:] [caminho]arquivo [parâmetros]

parâmetros      Especifica qualquer informação de linha de comando necessária ao programa em lotes.

Se as extensões de comando estiverem ativadas, CALL será alterado como a seguir:

O comando CALL agora aceita rótulos como destino. A sintaxe é:

CALL :rótulo argumentos

Um novo contexto de arquivo em lotes é criado com os argumentos especificados e o controle é transferido para a instrução após o rótulo especificado. Você deve "sair" duas vezes ao alcançar o final do arquivo de script em lotes duas vezes. Na primeira vez em que você ler o final, o controle retornará para logo após a instrução CALL. Na segunda vez, sairá do script em lotes. Digite GOTO /? para obter uma descrição da extensão GOTO :EOF que lhe permitirá "retornar" de um script em lotes.

Além disso, a expansão de referências de argumento de script em lotes (%0, %1, etc.) foram alteradas como a seguir:

%\* em um script em lotes refere-se a todos os argumentos (ex.: %1 %2 %3 %4 %5 ...)

A substituição de parâmetros em lotes (%n) foi aprimorada. Agora é possível usar a seguinte sintaxe opcional:

%~1	- expande %1 removendo quaisquer aspas ("")
%~f1	- expande %1 para um nome de caminho totalmente qualificado
%~d1	- expande %1 para somente uma letra de unidade
%~p1	- expande %1 para somente um caminho
%~n1	- expande %1 para somente um nome de arquivo
%~x1	- expande %1 para somente uma extensão de arquivo
%~s1	- o caminho expandido contém somente nomes curtos
%~a1	- expande %1 para atributos de arquivo
%~t1	- expande %1 para data/hora do arquivo
%~z1	- expande %1 para o tamanho do arquivo
%~\$PATH:1	- pesquisa as pastas listadas na variável de ambiente PATH e expande %1 para o nome totalmente qualificado da primeira encontrada. Se o nome da variável de ambiente não estiver definido ou o arquivo não for encontrado pela pesquisa, esse modificador será expandido para a cadeia de caracteres vazia

Os modificadores podem ser combinados para obter resultados compostos:

```
%~dp1      - expande %1 somente para uma letra da unidade e
caminho
%~nx1      - expande %1 somente para uma extensão e nome de
arquivo
%~dp$PATH:1 - pesquisa as pastas listadas na variável de ambiente
              PATH para %1 e expande para a
              letra da unidade e caminho da primeira encontrada.
%~ftza1     - expande %1 para uma linha de saída do tipo DIR
```

Nos exemplos acima %1 e PATH podem ser substituídos por outros valores válidos. A sintaxe %~ é finalizada por um número de argumento válido. Os modificadores %~ não podem ser usados com %\*

Exibe o nome da pasta ou altera a pasta atual.

```
CHDIR [/D] [unidade:] [caminho]  
CHDIR[...]  
CD [/D] [unidade:] [caminho]  
CD[...]
```

.. Especifica que você quer ir para a pasta pai.

Digite CD unidade: para exibir a pasta atual na unidade especificada.  
Digite CD sem parâmetros para exibir a unidade e pasta atuais.

Use a chave /D para alterar a unidade atual e a pasta atual dentro desta unidade.

Se as extensões de comando estiverem ativadas, o CHDIR será alterado como a seguir:

A cadeia de caracteres da pasta atual será convertida para usar a mesma diferenciação de maiúsculas e minúsculas que os nomes no disco. Portanto, CD C:\TEMP na verdade definiria a pasta atual como C:\Temp se estivesse assim no disco.

O comando CHDIR não trata espaços como delimitadores, portanto é possível usar

CD para um nome de subpasta que contenha um espaço sem colocar o nome entre aspas. Por exemplo:

```
cd \winnt\profiles\username\programs\start menu
```

é o mesmo que:

```
cd "\winnt\profiles\username\programs\start menu"
```

que é o que deveria ser digitado se as extensões estivessem desativadas.

Exibe ou define o n·ero da p·rtina de c·igo ativa.

CHCP [nnn]

nnn Especifica um n·ero de p·rtina de c·igo.

Digite CHCP sem par·metros para exibir o n·ero da p·rtina de c·igo ativa.

Exibe o nome da pasta ou altera a pasta atual.

```
CHDIR [/D] [unidade:] [caminho]  
CHDIR[...]  
CD [/D] [unidade:] [caminho]  
CD[...]
```

.. Especifica que você quer ir para a pasta pai.

Digite CD unidade: para exibir a pasta atual na unidade especificada.  
Digite CD sem parâmetros para exibir a unidade e pasta atuais.

Use a chave /D para alterar a unidade atual e a pasta atual dentro desta unidade.

Se as extensões de comando estiverem ativadas, o CHDIR será alterado como a seguir:

A cadeia de caracteres da pasta atual será convertida para usar a mesma diferenciação de maiúsculas e minúsculas que os nomes no disco. Portanto, CD C:\TEMP na verdade definiria a pasta atual como C:\Temp se estivesse assim no disco.

O comando CHDIR não trata espaços como delimitadores, portanto é possível usar

CD para um nome de subpasta que contenha um espaço sem colocar o nome entre aspas. Por exemplo:

```
cd \winnt\profiles\username\programs\start menu
```

é o mesmo que:

```
cd "\winnt\profiles\username\programs\start menu"
```

que é o que deveria ser digitado se as extensões estivessem desativadas.

Verifica um disco e exibe um relatório de status.

CHKDSK [volume[[caminho]nome\_de\_arquivo]] [/F] [/V] [/R] [/X] [/I] [/C] [/L[:tamanho]] [/B] [/scan] [/spotfix]

volume Especifica a letra da unidade (seguida de dois-pontos),  
o ponto de montagem ou o nome do volume.  
nome do arquivo Apenas FAT/FAT32: especifica os arquivos nos quais  
será verificada a fragmentação.  
/F Corrigir os erros do disco.  
/V Em FAT/FAT32: exibe o caminho completo e o nome de cada  
arquivo no disco.  
Em NTFS: exibe mensagens de limpeza, se houver.  
/R Localiza setores inválidos e recupera informações legíveis (implicar /F quando /scan não estiver especificado).  
Somente NTFS: altera o tamanho do arquivo de log para um número especificado de quilobytes. Se o tamanho não for especificado, exibir o tamanho atual.  
Força a desmontagem do volume primeiro, se necessário.  
/L:size para um não fornecer ser  
Todos os identificadores abertos no volume devem ser  
inválidos (implica /F).  
Somente NTFS: executa uma verificação menos rigorosa das entradas de índice.  
Somente NTFS: ignora a verificação de ciclos nas pastas.  
Somente NTFS: reavalia os clusters incorretos no volume (implica /R).  
Somente NTFS: executa uma verificação online no volume.  
Somente NTFS: (deve ser usado com "/scan") Ignora todo o reparo online; todos os defeitos encontrados são enfileirados para reparo offline (por exemplo, "chkdsk /spotfix").  
Somente NTFS: (deve ser usado com "/scan") Usa mais recursos do sistema para concluir o exame.

o negativo mais rápido possível. Isso pode ter um impacto no desempenho das outras tarefas em execução no sistema.

/spotfix Somente NTFS: executa a correção de ponto no volume

/sdcleanup Somente NTFS: coleta de lixo desnecessária para dados do descritor de segurança (implica /F).

/offline scanandfix Executa uma verificação offline e corrige no volume.

A opção /I ou /C reduz o tempo necessário para executar Chkdsk ignorando determinadas verificações do volume.

Exibe ou modifica a verificação de disco no momento da inicialização.

```
CHKNTFS volume [...]
CHKNTFS /D
CHKNTFS /T[:tempo]
CHKNTFS /X volume [...]
CHKNTFS /C volume [...]
```

volume	Especifica a letra da unidade (seguida de dois-pontos), o ponto de montagem ou o nome do volume.
/D	Restaura o comportamento padrão da máquina; todas as unidades serão verificadas no momento da inicialização e o CHKDSK é executado nas unidades sujas.
/T:tempo	Altera o tempo de contagem regressiva inicial de AUTOCHK para o tempo especificado em segundos. Se o tempo não
for	especificado, altera a configuração atual.
/X	Exclui uma unidade da verificação no momento da inicialização padrão. As unidades excluídas não serão acumuladas entre invocações de comando.
/C	Agenda a verificação de uma unidade para o momento da inicialização; o CHKDSK será executado se a unidade
estiver	suja.

Se nenhuma opção estiver especificada, o CHKNTFS informará se a unidade especificada está suja ou agendada para verificação na próxima inicialização.

Limpa a tela.

CLS

Inicia uma nova instância do interpretador de comando do Windows

CMD [/A | /U] [/Q] [/D] [/E:ON | /E:OFF] [/F:ON | /F:OFF] [/V:ON |  
/V:OFF]  
[[/S] [/C | /K] cadeia\_de\_caracteres]

/C Executa o comando especificado pela cadeia\_de\_caracteres e é encerrado  
/K Executa o comando especificado pela cadeia\_de\_caracteres mas permanece  
/S Modifica o tratamento da cadeia\_de\_caracteres após /C ou /K (ver abaixo)  
/Q Desativa o comando echo  
/D Desativa a execução de comandos AutoRun do Registro (ver abaixo)  
/A Faz com que a saída de comandos internos para um pipe ou arquivo seja em ANSI  
/U Faz com que a saída de comandos internos para um pipe ou arquivo seja em Unicode  
/T:fg Define as cores em primeiro/segundo plano (consulte COLOR /? para obter mais informações)  
/E:ON Ativa extensões de comando (ver abaixo)  
/E:OFF Desativa extensões de comando (ver abaixo)  
/F:ON Ativa caracteres de conclusão de nome de arquivo e de pasta (ver abaixo)  
/F:OFF Desativa caracteres de conclusão de nome de arquivo e de pasta (ver abaixo)  
/V:ON Ativa a expansão de variáveis de ambiente atrasada usando ! como delimitador. Por exemplo, /V:ON permite que !var! expanda a variável var no tempo de execução. A sintaxe var expande variáveis no momento da entrada, que é um procedimento bem diferente quando está dentro de um loop de FOR.  
/V:OFF Desativa a expansão de ambiente atrasada.

Observe que vários comandos separados pelo separador de comando '&&' são aceitos para cadeia de caracteres se estiverem entre aspas. Além disso,

por razões de compatibilidade, /X é o mesmo que /E:ON, /Y é o mesmo que /E:OFF e /R é o mesmo que /C. Qualquer outra opção é ignorada.

Se /C ou /K for especificado, o restante da linha de comando após a opção será processado como uma linha de comando, onde a seguinte lógica é usada para processar caracteres de aspas (""):

1. Se todas as condições a seguir forem atendidas, as aspas na linha de comando serão preservadas:
  - nenhuma opção /S
  - exatamente duas aspas
  - nenhum caractere especial entre as duas aspas, onde o especial é um dos seguintes: &<>()@^|
  - há um ou mais caracteres de espaço entre as duas aspas
  - a cadeia de caracteres entre as duas aspas é o nome de um arquivo executável.
2. Caso contrário, o costume é ver se o primeiro caractere é um caractere de aspas e, se for, retirar o primeiro caractere e

remover o último caractere de aspas na linha de comando,  
preservando  
qualquer texto após as últimas aspas.

Se /D NÃO estiver especificado na linha de comando, quando o CMD.EXE for iniciado, ele procurará as variáveis de Registro REG\_SZ/REG\_EXPAND\_SZ a seguir e, se nenhuma ou ambas estiverem presentes, serão executadas primeiro.

HKEY\_LOCAL\_MACHINE\Software\Microsoft\Command Processor\AutoRun  
e/ou

HKEY\_CURRENT\_USER\Software\Microsoft\Command Processor\AutoRun

As Extensões de Comando estão ativadas por padrão. Você também pode desabilitar as extensões de uma determinada invocação usando a opção /E:OFF.

Para habilitar ou desabilitar as extensões de todas as invocações do CMD.EXE em uma máquina e/ou sessão de logon de usuário, configure um dos valores REG\_DWORD a seguir, ou ambos os valores, no Registro usando o REGEDIT.EXE:

HKEY\_LOCAL\_MACHINE\Software\Microsoft\Command Processor\EnableExtensions  
e/ou

HKEY\_CURRENT\_USER\Software\Microsoft\Command Processor\EnableExtensions

para 0x1 ou 0x0. A configuração específica do usuário tem precedência sobre a configuração do computador. As opções da linha de comando têm precedência sobre as configurações do Registro.

Em um arquivo em lotes, os argumentos SETLOCAL ENABLEEXTENSIONS ou DISABLEEXTENSIONS têm precedência sobre a opção /E:ON ou /E:OFF. Consulte SETLOCAL /? para obter detalhes.

As extensões de comando envolvem alterações e/ou adições nos comandos a seguir:

DEL ou ERASE  
COLOR  
CD ou CHDIR  
MD ou MKDIR  
PROMPT  
PUSHD  
POPD  
SET  
SETLOCAL  
ENDLOCAL  
IF  
FOR  
CALL  
SHIFT  
GOTO

START (também inclui as alterações feitas na invocação de comando externo)  
ASSOC  
FTYPE

Para obter detalhes específicos, digite commandname /? para exibir os detalhes.

A expansão de variáveis de ambiente atrasada NÃO é ativada por padrão. É possível habilitar ou desabilitar a expansão de variáveis de ambiente atrasada para uma determinada invocação do CMD.EXE com a opção /V:ON ou /V:OFF. Para habilitar ou desabilitar as extensões atrasadas de todas as invocações do CMD.EXE em uma máquina e/ou sessão de logon de usuário, configure um dos valores REG\_DWORD a seguir, ou ambos os valores, no Registro usando o REGEDIT.EXE:

HKEY\_LOCAL\_MACHINE\Software\Microsoft\Command Processor\DelayedExpansion

e/ou

HKEY\_CURRENT\_USER\Software\Microsoft\Command Processor\DelayedExpansion

para 0x1 ou 0x0. A configuração específica do usuário tem precedência sobre a configuração do computador. As opções da linha de comando têm precedência sobre as configurações do Registro.

Em um arquivo em lotes, os argumentos ENABLEDELAYEXPANSION ou DISABLEDELAYEXPANSION têm precedência sobre a opção /V:ON ou /V:OFF. Consulte SETLOCAL /? para obter detalhes.

Se a expansão de variáveis de ambiente atrasada estiver habilitada, o caractere de exclamação poderá ser usado para substituir o valor de uma variável de ambiente no tempo de execução.

Você pode ativar ou desativar o preenchimento para uma chamada específica do CMD.EXE com a opção /F:ON ou /F:OFF. Você pode ativar ou desativar o preenchimento para todas as chamadas do CMD.EXE em um computador e/ou sessão de logon de usuário definindo qualquer um dos valores REG\_DWORD a seguir (ou ambos) no Registro usando REGEDIT.EXE:

HKEY\_LOCAL\_MACHINE\Software\Microsoft\Command Processor\CompletionChar  
HKEY\_LOCAL\_MACHINE\Software\Microsoft\Command Processor\PathCompletionChar

e/ou

HKEY\_CURRENT\_USER\Software\Microsoft\Command Processor\CompletionChar  
HKEY\_CURRENT\_USER\Software\Microsoft\Command Processor\PathCompletionChar

com o valor hexadecimal de um caractere de controle a ser usado para determinada função (ex.: 0x4 é Ctrl-D e 0x6 é Ctrl-F). As configurações específicas do usuário têm precedência sobre as configurações do computador.

As opções da linha de comando têm precedência sobre as configurações do Registro.

Se a conclusão for ativada com a opção /F:ON, os dois caracteres de controle usados serão Ctrl-D para a conclusão de nome de pasta e Ctrl-F para a conclusão de nome de arquivo. Para desativar um determinado caractere de conclusão no Registro, use o valor do espaço (0x20), que não é um caractere de controle válido.

A conclusão é chamada quando você digita um dos dois caracteres de controle.

A função de conclusão usa a cadeia de caracteres do caminho à esquerda do cursor, acrescenta um caractere curinga a ela, se já não existir um, e cria

uma lista de caminhos correspondentes. Em seguida, exibe o primeiro caminho

correspondente. Se nenhum caminho corresponder, ela emite um aviso e não altera nada. Depois, o pressionamento repetido do mesmo caractere de controle

percorrerá a lista de caminhos correspondentes. O pressionamento da tecla Shift com o caractere de controle permite percorrer a lista de trás para a frente. Se você fizer qualquer edição na linha e pressionar o caractere de

controle novamente, a lista salva de caminhos correspondentes é descartada e

uma nova é gerada. O mesmo ocorre se você alternar entre a conclusão de nome

de arquivo e de pasta. A única diferença entre os dois caracteres de controle

é que o caractere de conclusão de arquivo corresponde a ambos os nomes de arquivo e de pasta, enquanto que o caractere de conclusão de pasta somente

corresponde a nomes de pastas. Se a conclusão de arquivo for usada em qualquer

um dos comandos de pasta internos (CD, MD ou RD), a conclusão de pasta será usada.

O código de conclusão trata corretamente de nomes de arquivos que contêm espaços ou outros caracteres especiais colocando aspas em volta do caminho

correspondente. Além disso, se você retornar com o cursor e chamar a conclusão

a partir de uma linha, o texto à direita do cursor no ponto em que a conclusão

foi chamada será descartado.

Os caracteres especiais que exigem aspas são:

<espaço>  
& () [] {}^=; !'+, `~

Configura as cores padrão de primeiro plano e tela de fundo do console.

COLOR [attr]

attr Especifica os atributos de cor da saída do console

Atributos de cor são especificados por DOIS dígitos hexadecimais. O primeiro corresponde à cor de tela de fundo; o segundo à cor de primeiro plano. Cada dígito pode ter apenas um dos seguintes valores:

0 = Preto	8 = Cinza
1 = Azul	9 = Azul claro
2 = Verde	A = Verde claro
3 = Verde-água	B = Verde-água claro
4 = Vermelho	C = Vermelho claro
5 = Roxo	D = Lilás
6 = Amarelo	E = Amarelo claro
7 = Branco	F = Branco brilhante

Caso nenhum argumento seja passado, este comando restaurará a cor de antes do CMD.EXE ser executado. Este valor vem ou da janela atual do console, ou da opção /T da linha de comando, ou do valor de DefaultColor no Registro.

O comando COLOR altera ERRORLEVEL para 1 se for tentado se executar o comando COLOR com as mesmas cores de primeiro plano e de tela de fundo.

Exemplo: "COLOR fc" gera o vermelho claro na tela de fundo branca brilhante

Compara o conteúdo de dois arquivos ou conjuntos de arquivos.

COMP [dados1] [dados2] [/D] [/A] [/L] [/N=número] [/C] [/OFF[LINE]]

dados1 Especifica o local e o(s) nome(s) do(s) primeiro(s)  
arquivo(s) para compará-lo.  
dados2 Especifica o local e o(s) nome(s) do(s) segundo(s)  
arquivo(s) para compará-lo.  
/D Exibe diferenças no formato decimal.  
/A Exibe diferenças em caracteres ASCII.  
/L Exibe números de linha para diferenças.  
/N=número Compara somente o primeiro número especificado de linhas em  
cada arquivo.  
/C Ignora a diferença de maiusculas e minúsculas nas  
letras ASCII ao comparar arquivos.  
/OFF[LINE] Não ignora arquivos com conjunto de atributos offline.

Para comparar conjuntos de arquivos, use caracteres curingas nos  
parâmetros  
dados1 e dados2.

Exibe ou altera a compacta#o de arquivos em parti#aes NTFS.

COMPACT [/C | /U] [/S[:pasta]] [/A] [/I] [/F] [/Q] [arquivo [...]]

/C            Compacta os arquivos especificados. Os arquivos que forem adicionados depois nesta pasta ser#o compactados.  
/U            Descompacta os arquivos especificados. Os arquivos que forem adicionados depois nesta pasta n#o ser#o compactados.  
/S            Efetua a opera#o desejada nos arquivos da pasta especificada e em todas as subpastas. Caso n#o seja especificado, ser usada a pasta atual.  
/A            Exibe os arquivos ocultos e de sistema. Normalmente, estes arquivos seriam omitidos.  
/I            Continua a executar a opera#o especificada mesmo ap#s a ocorr^ncia de erros. Normalmente, o programa pararia quando um erro fosse encontrado.  
/F            For#a a opera#o de compacta#o em todos os arquivos especificados, mesmo naqueles que j# estejam compactados. Normalmente, os arquivos j# compactados seriam ignorados.  
/Q            Relata somente as informa#aes essenciais.  
arquivo      Especifica um padr#o, arquivo ou pasta.

Utilizado sem parfmetros, COMPACT exibe o estado de compacta#o da pasta atual e de quaisquer arquivos nela contidos. Voc^ pode usar m#ltiplos nomes de arquivos e caracteres curingas, e deve colocar espacos entre os parfmetros.

Converte volumes FAT em NTFS.

```
CONVERT volume /FS:NTFS [/V] [/CvtArea:nome_de_arquivo] [/NoSecurity]  
[/X]
```

volume Especifica a letra da unidade (seguida de dois-pontos), ponto de montagem ou nome do volume.  
/FS:NTFS Especifica a conversão do volume em NTFS.  
/V Especifica que Convert deve ser executado no modo detalhado.  
/CvtArea:nome\_de\_arquivo Especifica um arquivo contendo no diretório raiz que será o espaço reservado para arquivos de sistema NTFS.  
/NoSecurity Especifica as configurações de segurança de arquivos e diretórios convertidos de modo que sejam acessíveis por todos os usuários.  
/X Força o volume a ser desmontado primeiro, caso necessário. Todos os identificadores abertos para o volume passam a ser invalidados.

Copia um ou mais arquivos para outro local.

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

origem Especifica o arquivo ou arquivos a serem copiados.  
/A Indica um arquivo de texto ASCII.  
/B Indica um arquivo binário.  
/D Permite que o arquivo de destino seja criado

descriptografado

destino Especifica o diretório e/ou nome de arquivo para os novos arquivos.  
/V Verifica se os novos arquivos são gravados corretamente.  
/N Usa um nome de arquivo curto, se disponível, ao copiar um arquivo com nome que não esteja em formato 8.3.  
/Y Suprime o prompt para você confirmar se deseja substituir um arquivo de destino existente.  
/-Y Exibe o prompt para você confirmar se deseja substituir um arquivo de destino existente.  
/Z Copia arquivos de rede no modo reiniciável.  
/L Se a origem for um vínculo simbólico, copie o vínculo para  
o destino em vez do arquivo real para o qual o vínculo de  
origem aponta.

A opção /Y pode ser predefinida na variável de ambiente COPYCMD. Pode ser substituída por /-Y na linha de comando. O padrão é solicitar durante a substituição, a menos que o comando COPY esteja sendo executado a partir de um script em lote.

Para acrescentar arquivos, especifique um único arquivo para destino, mas vários arquivos para origem (usando caracteres curinga ou o formato arquivo1+arquivo2+arquivo3).

Exibe ou define a data.

DATE [/T | data]

Digite DATE sem parâmetros para exibir a data atual e poder digitar a nova data. Pressione ENTER para manter a data inalterada.

Se as extensões de comando estiverem ativadas, o comando DATE dá suporte para a opção /T, que o instrui a exibir apenas a data atual, sem solicitar uma nova data.

Exclui um ou mais arquivos.

DEL [/P] [/F] [/S] [/Q] [/A[[::]atributos]] nomes  
ERASE [/P] [/F] [/S] [/Q] [/A[[::]atributos]] nomes

nomes Especifica uma lista de um ou mais arquivos ou pastas.  
Caracteres curinga podem ser usados para excluir vários  
arquivos. Se uma pasta for especificada, todos os  
arquivos dentro dela serão excluídos.

/P Solicita confirmação antes de excluir cada arquivo.  
/F Força a exclusão de arquivos somente leitura.  
/S Exclui arquivos especificados de todas as subpastas.  
/Q Modo silencioso, não pede confirmação para excluir com  
caractere curinga global  
/A Seleciona arquivos a serem excluídos com base nos  
atributos atributos R Arquivos somente leitura S Arquivos do sistema  
H Arquivos ocultos A Arquivos prontos  
I Arquivos sem conteúdo indexado L Pontos de nova  
análise - Prefixo significando negação

Se as extensões de comando estiverem ativadas, os comandos DEL e ERASE  
serão  
alterados como a seguir:

A semântica de exibição da opção /S é revertida ao mostrar  
somente os arquivos excluídos, e não os que não pôde localizar.

Exibe uma lista de arquivos e subdiretórios em um diretório.

```
DIR [unidade:] [caminho] [arquivo] [/A[[:atributos]] [/B] [/C] [/D] [/L]
      [/N]
      [/O[[:ordem_de_classificação]] [/P] [/Q] [/R] [/S]
      [/T[[:campo_de_tempo]]]
      [/W] [/X] [/4]
```

```
[unidade:] [caminho] [nome_de_arquivo]
          Especifica a unidade, o diretório e/ou arquivos a serem
listados.
```

/A	Exibe arquivos com atributos especificados.
atributos	D Diretórios R Arquivos somente leitura H Arquivos ocultos A Arquivos prontos para arquivamento S Arquivos de sistema I Arquivos sem conteúdo indexado L Pontos de nova análise - Prefixo significando negação /B Usa formatação básica (sem informações de cabeçalho ou resumo). /C Exibe o separador de milhar em tamanhos de arquivos. É o padrão. Use /-C para desabilitar a exibição do separador. /D O mesmo que amplo, mas os arquivos são classificados na lista por coluna. /L Usa letras minúsculas. /N Novo formato de lista longo onde os nomes de arquivos estão à extrema direita. /O Lista por arquivos na ordem classificada. ordem_de_classificação N Por nome (alfabético) S Por tamanho (menor primeiro) E Por extensão (alfabética) D Por data/hora(mais antiga primeiro) primeiro) G Grupo de diretórios primeiro - Prefixo para inverter a ordem /P Pausa após cada tela de informações. /Q Exibe o proprietário do arquivo. /R Exibe fluxos de dados alternados do arquivo. /S Exibe os arquivos no diretório especificado e em todos os subdiretórios. /T Controla qual campo de tempo é exibido ou usado na classificação campo_de_tempo C Criação A Último Acesso W Última Gravação /W Usa o formato de lista amplo. /X Exibe os nomes curtos gerados para nomes de arquivos diferentes do formato 8.3. O formato é /N com o nome curto inserido antes do nome longo. Se nenhum nome curto estiver presente, serão exibidos espaços no seu lugar. /4 Exibe anos de quatro dígitos

As opções podem estar predefinidas na variável de ambiente DIRCMD.  
Substituir nas opções predefinidas ao prefixar qualquer opção  
com - (hífen)--por exemplo, /-W.

Compara o conteúdo de dois disquetes.

**DISKCOMP** [unidade1: [unidade2:]]

Copia o conteúdo de um disquete para outro.

DISKCOPY [unidade1: [unidade2:]] [/V]

/V Verifica se as informações são copiadas corretamente.

Os dois disquetes devem ser do mesmo tipo.

Você pode especificar a mesma unidade para unidade1 e unidade2.

Edita linhas de comando, recupera comandos do Windows e cria macros.

```
DOSKEY [/REINSTALL] [/LISTSIZE=tamanho] [/MACROS[:ALL | :execut vel]]  
[/HISTORY] [/INSERT | /OVERSTRIKE] [/EXENAME=execut vel]  
[/MACROFILE=arquivo] [nome_macro=[texto]]  
  
/REINSTALL           Instala uma nova c pia de Doskey.  
/LISTSIZE=tamanho    Define o tamanho do buffer de hist rico de  
comandos.  
/MACROS              Exibe todas as macros de Doskey.  
/MACROS:ALL          Exibe todas as macros de Doskey de todos os  
execut veis que tenham macros de Doskey.  
/MACROS:execut vel   Exibe todas as macros de Doskey do execut vel.  
/HISTORY             Exibe todos os comandos armazenados na mem ria.  
/INSERT               Especifica que o novo texto , inserido no texto  
anterior.  
/OVERSTRIKE          Especifica que o novo texto substitui o texto  
anterior.  
/EXENAME=execut vel  Especifica o execut vel.  
/MACROFILE=arquivo   Especifica um arquivo de macros para instalar.  
nome_macro           Especifica o nome de uma macro criada.  
texto                Especifica comandos que voc^ deseja gravar.
```

Setas PARA CIMA e PARA BAIXO recuperam comandos; ESC limpa a linha de comando;  
F7 exibe o hist rico de comandos; ALT+F7 limpa o hist rico de comandos;  
F8 pesquisa o hist rico de comandos;  
F9 seleciona um comando por n mero; ALT+F10 limpa as defini es de macros.

A seguinte codifica o especial , usada nas defini es de macros de Doskey:

\$T Separador de comandos. Permite m ltiplos comandos em uma macro.  
\$1-\$9 Parfmetros em lotes. Equivalente a %1-%9 nos programas em lotes.  
\$\* Substitu;do pelo que segue o nome da macro na linha de comando.

```
DRIVERQUERY [/S sistema [/U nome_usuário [/P [senha]]]]  
           [/FO formato] [/NH] [/SI] [/V]
```

**Descrição:**

Permite que um administrador exiba uma lista de drivers de dispositivo instalados.

**Lista de parâmetros:**

/S	sistema	Especifica o sistema remoto ao qual se conectar.
/U	[domínio\]usuário	Especifica o contexto de usuário em que o comando deve ser executado.
/P	[senha]	Especifica a senha para contexto de usuário.
/FO	formato	Especifica o tipo de saída para exibição. Valores válidos a serem passados com opção são "TABLE", "LIST", "CSV".
/NH		Especifica que o "cabeçalho de coluna" não deve ser exibido. Válido para formatos "TABLE" e "CSV" somente.
/SI		Fornece informações sobre drivers assinados.
/V		Exibe a saída detalhada. Não é válido para drivers assinados.
/?		Exibe esta mensagem de ajuda.

**Exemplos:**

```
DRIVERQUERY  
DRIVERQUERY /FO CSV /SI  
DRIVERQUERY /NH  
DRIVERQUERY /S endereço_ip /U usuário /V  
DRIVERQUERY /S sistema /U domínio\usuário /P senha /FO LIST
```

Exibe mensagens ou ativa ou desativa o eco de comando.

ECHO [ON | OFF]  
ECHO [mensagem]

Digite ECHO sem parâmetros para exibir a configuração do eco atual.

Termina a localização das alterações de ambiente em um arquivo em lotes. As alterações de ambiente feitas após ENDLOCAL ser ativado não são específicas do arquivo em lotes; as configurações anteriores não serão restauradas na finalização do arquivo em lotes.

ENDLOCAL

Se as extensões de comando estiverem ativadas, o ENDLOCAL será alterado como  
a seguir:

Se o SETLOCAL correspondente ativar ou desativar as extensões de comando usando as novas opções ENABLEEXTENSIONS ou DISABLEEXTENSIONS, após o ENDLOCAL, o estado ativado/desativado das extensões de comando voltará ao que costumava ser antes de corresponder à execução do comando SETLOCAL.

Exclui um ou mais arquivos.

DEL [/P] [/F] [/S] [/Q] [/A[[::]atributos]] nomes  
ERASE [/P] [/F] [/S] [/Q] [/A[[::]atributos]] nomes

nomes Especifica uma lista de um ou mais arquivos ou pastas.  
Caracteres curinga podem ser usados para excluir vários  
arquivos. Se uma pasta for especificada, todos os  
arquivos dentro dela serão excluídos.

/P Solicita confirmação antes de excluir cada arquivo.  
/F Força a exclusão de arquivos somente leitura.  
/S Exclui arquivos especificados de todas as subpastas.  
/Q Modo silencioso, não pede confirmação para excluir com  
caractere curinga global  
/A Seleciona arquivos a serem excluídos com base nos  
atributos atributos R Arquivos somente leitura S Arquivos do sistema  
H Arquivos ocultos A Arquivos prontos  
I Arquivos sem conteúdo indexado L Pontos de nova  
análise - Prefixo significando negação

Se as extensões de comando estiverem ativadas, os comandos DEL e ERASE  
serão  
alterados como a seguir:

A semântica de exibição da opção /S é revertida ao mostrar  
somente os arquivos excluídos, e não os que não pôde localizar.

Encerra o programa CMD.EXE (interpretador de comando) ou o script em lote atual.

EXIT [/B] [Código\_de\_saída]

/B Especifica a saída do script em lote atual em vez do CMD.EXE. Se for executado de fora de um script em lote,  
sairá do CMD.EXE

Código\_de\_saída Especifica um número. Se /B for especificado, definirá ERRORLEVEL como esse número. Se sair do CMD.EXE,  
definirá o código de saída do processo como esse número.

Compara dois arquivos ou conjuntos de arquivos e exibe as diferenças entre eles

FC [/A] [/C] [/L] [/LBn] [/N] [/OFF[LINE]] [/T] [/U] [/W] [/nnnn]  
[unidade1:] [caminho1] arquivo1 [unidade2:] [caminho2] arquivo2  
FC /B [unidade1:] [caminho1] arquivo1 [unidade2:] [caminho2] arquivo2

/A Exibe apenas a primeira e última linhas de cada conjunto de diferenças.  
/B Executa uma comparação binária.  
/C Ignora maiúsculas e minúsculas.  
/L Compara arquivos como texto ASCII.  
/LBn Define o número máximo de incompatibilidades consecutivas em relativo ao número de linhas especificado.  
/N Exibe os números de linha em uma comparação ASCII.  
/OFF[LINE] Não ignora arquivos com conjunto de atributos offline.  
/T Não expande tabulações em espaços.  
/U Compara arquivos como arquivos de texto UNICODE.  
/W Compacta espaços em branco (tabulações e espaços) para comparação.  
/nnnn Especifica o número de linhas consecutivas que devem coincidir depois de uma incompatibilidade.  
[unidade1:] [caminho1] arquivo1 Especifica o primeiro arquivo ou conjunto de arquivos a comparar.  
[unidade2:] [caminho2] arquivo2 Especifica o segundo arquivo ou conjunto de arquivos a comparar.

Localiza uma cadeia de caracteres de texto em um ou mais arquivos.

FIND [/V] [/C] [/N] [/I] [/OFF[LINE]] "cadeia de caracteres" [[unidade:]  
[caminho]arquivo[ ...]]

/V	Exibe todas as linhas que NÇO contêm a cadeia de caracteres
	especificada.
/C	Exibe apenas o número de linhas que contêm a cadeia de caracteres.
/N	Exibe o número de linha de cada linha.
/I	Ignora maiúsculas/minúsculas ao localizar uma cadeia de caracteres.
/OFF[LINE]	NÃO ignora arquivos com conjunto de atributos off-line.
"string"	Especifica a cadeia de caracteres de texto a ser localizada.
[unidade:] [caminho]arquivo	Especifica os arquivos a serem localizados.

Se um caminho não for especificado, FIND localiza o texto digitado no  
prompt  
ou obtido de outro comando.

Procura cadeias de caracteres em arquivos.

FINDSTR [/B] [/E] [/L] [/R] [/S] [/I] [/X] [/V] [/N] [/M] [/O] [/P]  
[/F:arquivo] [/C:cadeia de caracteres] [/G:arquivo]  
[/D:lista\_de\_pastas] [/A:atributos\_de\_cor] [/OFF[LINE]]  
cadeia de caracteres [[unidade:] [caminho]arquivo[ ...]]

/B Corresponde ao padrão se estiver no inicio  
de uma linha.  
/E Corresponde ao padrão se estiver no final  
de uma linha.  
/L Usa cadeias de caracteres de pesquisa  
literalmente.  
/R Usa cadeias de caracteres de pesquisa como  
expressões regulares.  
/S Procura por arquivos correspondentes na  
pasta atual e em todas as subpastas.  
/I Especifica que a pesquisa não diferenciar  
maiúsculas de minúsculas.  
/X Imprime as linhas que têm correspondência  
exata.  
/V Imprime somente as linhas que não contêm  
uma correspondência.  
/N Imprime o número da linha antes de cada  
linha correspondente.  
/M Imprime somente o nome do arquivo se um  
arquivo contiver uma correspondência.  
/O Imprime o deslocamento de caractere antes  
de cada linha correspondente.  
/P Ignora arquivos com caracteres não  
imprimíveis.  
/OFF[LINE] Não ignora caracteres com conjunto de  
atributos offline.  
/A:atributos Especifica atributo de cor com dois dígitos  
hexadecimais.  
Consulte "color /?"  
/F:arquivo Lê a lista de arquivos do arquivo  
especificado (/ significa console).  
/C:cadeia de caracteres Usa uma cadeia de caracteres especificada  
como uma cadeia de caracteres de pesquisa  
literal.  
/G:arquivo Obtém cadeias de caracteres de pesquisa do  
arquivo especificado (/ significa console).  
/D:lista\_de\_pastas Procura uma lista de pastas delimitada por  
ponto-e-vírgula  
cadeias de caracteres Texto a ser procurado.  
[unidade:] [caminho]nome\_de\_arquivo Especifica um ou mais arquivos a procurar.

Use espaços para separar várias cadeias de caracteres de pesquisa, a  
menos

que o argumento seja antecedido por /C. Por exemplo, 'FINDSTR "bom dia"  
x.y'  
procura por "bom" ou "dia" no arquivo x.y. 'FINDSTR /C:"bom dia" x.y'  
procura por "bom dia" no arquivo x.y.

Refer^ncia r pida de expressäes regulares:

.	Curinga: qualquer caractere
*	Repetir: zero ou mais ocorr^ncias do caractere anterior ou classe
^	Posit^o na linha: inicio da linha
\$	Posit^o na linha: final da linha
[class]	Classe de caractere: qualquer caractere no conjunto
[^class]	Classe inversa: qualquer caractere que n^o esteja no conjunto
[x-y]	Intervalo: qualquer caractere dentro do intervalo especificado
\x	Escape: uso literal de metacaractere x
\<xyz	Posit^o na palavra: inicio da palavra
xyz\>	Posit^o na palavra: final da palavra

Para obter informa^aes detalhadas sobre as expressäes regulares de FINDSTR,  
consulte a Refer^ncia de comandos online.

Executa um comando especificado para cada arquivo em um conjunto de arquivos.

FOR %variável IN (conjunto) DO comando [parâmetros]

%variável Especifica um parâmetro substituível formado por uma só letra.  
(conjunto) Especifica um conjunto de um ou mais arquivos. Podem-se usar curingas.  
comando Especifica o comando a ser executado em cada arquivo.  
parâmetros Especifica os parâmetros ou opções do comando especificado.

Para usar o comando FOR em um programa em lotes, especifique %%variável em vez de %variável. Diferenciam-se maiúsculas de minúsculas nos nomes das variáveis, de forma que, por exemplo, %i é diferente de %I.

Se as extensões de comando estiverem ativadas, haverá suporte para os formatos adicionais do comando FOR

FOR /D %variável IN (conjunto) DO comando [parâmetros]

Se o conjunto contiver curingas, ele corresponderá aos nomes de pasta em vez de nomes de arquivo.

FOR /R [[unidade:]caminho] %variável IN (conjunto) DO comando [comando-parâmetros]

Orienta a árvore de diretórios com raiz na [unidade:]caminho, executando a instrução FOR em cada diretório da árvore. Se não houver especificação de diretório após /R, o diretório atual será pressuposto.

Se o conjunto for apenas um único caractere de ponto (.), ele enumerará apenas a árvore de diretórios.

FOR /L %variável IN (início,incremento,fim) DO comando [comando-parâmetros]

O conjunto é uma sequência de números de início ao fim, por quantidade de incremento. Portanto, (1,1,5) geraria a sequência 1 2 3 4 5 e (5,-1,1) geraria a sequência (5 4 3 2 1)

FOR /F ["opções"] %variável IN (conjunto\_de\_arquivos) DO comando [comando-parâmetros]

FOR /F ["opções"] %variável IN ("cadeia\_de\_caracteres") DO comando [comando-parâmetros]

FOR /F ["opções"] %variável IN ('comando') DO comando [comando-parâmetros]

ou, se a opção usebackq estiver presente:

FOR /F ["opções"] %variável IN (conjunto\_de\_arquivos) DO comando [comando-parâmetros]

FOR /F ["opções"] %variável IN ('cadeia\_de\_caracteres') DO comando [comando-parâmetros]

```
FOR /F ["opções"] %variável IN (`comando`) DO comando [comando-parâmetros]
```

conjunto de nomes de arquivo é um ou mais nomes de arquivos. Cada arquivo é aberto, lido e processado antes de se passar para o próximo arquivo em conjunto de nomes de arquivo. O processamento consiste em ler

o arquivo, quebrá-lo em linhas individuais de texto e analisar cada linha

para zero ou mais tokens. O corpo de for loop é chamado em seguida com o

conjunto de valores de variável para a(s) cadeia(s) de caracteres de token encontrada(s). Por padrão, /F transfere o primeiro token separado

por espaço de cada linha de cada arquivo. As linhas em branco são ignoradas. É possível substituir o comportamento de análise padrão especificando os parâmetros "opções" opcionais. Essa é uma cadeia de caracteres entre aspas que contém uma ou mais palavras-chave para especificar diferentes opções de análise. As palavras-chave são:

eol=c	- especifica caractere de comentário de fim de linha
	(somente um)
skip=n	- especifica o número de linhas a serem ignorados no
	início do arquivo.
delims=xxx	- especifica um conjunto de delimitadores.
Substitui	o conjunto de delimitadores padrão de espaço e tabulação.
tokens=x,y,m-n	- especifica quais tokens de cada linha serão transferidos para o corpo de cada iteração.
um	Isso alocará nomes de variáveis adicionais. O formato m-n é um intervalo, especificando do emésimo até o enésimo tokens. Se o último caractere na cadeia de caracteres tokens= for asterisco, uma variável adicional será alocada e receberá o texto restante na linha após o último token analisado.
usebackq	- especifica que a nova semântica está em vigor, onde uma cadeia de caracteres entre aspas é executada como um comando e uma cadeia de caracteres entre aspas simples é uma comando de cadeia de caracteres literal e permite o uso de aspas para citar nomes de arquivos em um conjunto de nomes de arquivo.

Alguns exemplos podem ajudar:

```
FOR /F "eol=; tokens=2,3* delims=, " %i in (meu_arquivo.txt) do @echo %i %j %k
```

analisaria cada linha em meu\_arquivo.txt, ignorando linhas que começassem

com um ponto-e-vírgula, passando o segundo e terceiro token de cada

linha para o corpo do comando for, com tokens delimitados por vírgulas

e/ou espaços. Observe que comandos dentro do corpo do for fazem referência a %i para obter o segundo token, %j para obter o terceiro token e %k para obter todos os tokens restantes após o terceiro. No caso

de nomes de arquivos contendo espaços, é necessário colocá-los entre aspas. Para usar aspas dessa forma, também é necessário usar a opção usebackq ou as aspas serão interpretadas como se estivessem definindo uma cadeia de caracteres literal a ser analisada.

%i é declarado explicitamente na instrução for e %j e %k são declarados implicitamente através da opção tokens=. É possível especificar até 26 tokens através da linha tokens=, contanto que isso

não origine uma tentativa de declarar uma variável mais alta do que a letra 'z' ou 'Z'. Lembre-se de que as variáveis FOR são formadas por uma

só letra, diferenciam maiúsculas e minúsculas, são globais e não é possível haver mais de 52 ativas no total em momento algum.

Também é possível usar a lógica de análise de FOR /F em uma cadeia de caracteres imediata, ao transformar o conjunto de arquivo dentro de parênteses em uma cadeia de caracteres entre aspas simples. A cadeia de caracteres será tratada como uma única linha de entrada de um arquivo e será analisada.

Finalmente, é possível usar o comando FOR /F para analisar a saída de um comando. Para isso, transforme o conjunto de arquivo dentro dos parênteses em uma cadeia de caracteres entre aspas simples. A cadeia de caracteres será tratada como uma linha de comando, que será transferida

para um CMD.EXE filho e a saída será capturada na memória e analisada como se fosse um arquivo. Assim, o seguinte exemplo:

```
FOR /F "usebackq delims==" %i IN (`conjunto`) DO @echo %i
```

enumeraria os nomes de variáveis de ambiente no ambiente atual.

Além disso, a substituição das referências da variável FOR foi aprimorada.

Agora é possível usar a seguinte sintaxe opcional:

%~I	- expande %I removendo quaisquer aspas em volta ("")
%~fI	- expande %I para um nome de caminho totalmente qualificado
%~dI	- expande %I para somente uma letra de unidade
%~pI	- expande %I para somente um caminho
%~nI	- expande %I para somente um nome de arquivo
%~xI	- expande %I para somente uma extensão de arquivo
%~sI	- o caminho expandido contém somente nomes curtos
%~aI	- expande %I para atributos de arquivo do arquivo
%~tI	- expande %I para data/hora do arquivo
%~zI	- expande %I para o tamanho do arquivo

%~\$PATH:I - pesquisa as pastas listadas na variável de ambiente PATH e expande %I para o nome totalmente qualificado da primeira pasta encontrada.

Se o nome da variável de ambiente não estiver definido ou o arquivo não for encontrado pela pesquisa, esse modificador expandirá para a cadeia de caracteres vazia

Os modificadores podem ser combinados para obter resultados compostos:

%~dpI	- expande %I para somente uma letra de unidade e caminho
%~nxI	- expande %I para somente um nome de arquivo e extensão
%~fsI	- expande %I para somente um nome de caminho completo com nomes curtos
%~dp\$PATH:I	- pesquisa as pastas listadas na variável de ambiente PATH para %I e expande para a letra de unidade e caminho da primeira encontrada.
%~ftzaI	- expande %I para uma linha de saída parecida com DIR

Nos exemplos acima, %I e PATH podem ser substituídos por outros valores válidos. A sintaxe %~ é terminada por um nome de variável FOR válido. O uso de nomes de variáveis em maiúsculas como %I facilita a leitura e evita confusão com os modificadores, que não fazem diferenciação entre maiúsculas e minúsculas.

Formata um disco para ser utilizado com o Windows.

```
FORMAT volume [/FS:sistema de arquivos] [/V:r&lt;ulo] [/Q] [/L]
[/:tamanho] [/C] [/I:estado] [/X] [/P:etapas] [/S:estado]
FORMAT volume [/V:r&lt;ulo] [/Q] [/F:tamanho] [/P:etapas]
FORMAT volume: [/V:r&lt;ulo] [/Q] [/T:trilhas /N:setores] [/P:etapas]
FORMAT volume: [/V:r&lt;ulo] [/Q] [/P:etapas]
FORMAT volume [/Q]

volume           Especifica a letra da unidade (seguida de dois-pontos),
                o ponto de montagem ou o nome do volume.
/FS:sistema_de_arquivos Especifica o tipo do sistema de arquivos
                           (FAT, FAT32, exFAT, NTFS, UDF).
/V:r&lt;ulo yyyy-yyyyy Especifica o r&lt;ulo do volume.
/Q               Executa uma formata&Eo r pida. Esta op&Eo substitui /P.
/C               Apenas NTFS: os arquivos criados no volume novo ser&Eo
                  compactados por padr&Eo.
/X               Força primeiro a desmontagem do volume, se necess rio.
                  Todos os identificadores abertos do volume n&Eo seriam
                  mais v lidos.
/R:revis&Eo       Apenas UDF: força a formata&Eo como uma vers&Eo de UDF
                  especifica (1.02, 1.50, 2.00, 2.01, 2.50). A revis&Eo
                  padr&Eo , 2.01.
/D               Apenas UDF 2.50: os metadados ser&Eo duplicados.
/L               Somente NTFS: usa os registros de arquivo de tamanho
                  grande.
arquivo          Por padr&Eo, o volume ser   formatado com registros do
                  tamanho pequeno.
/A:tamanho       Substitui o tamanho padr&Eo da unidade de aloca&Eo. Para
                  uso geral, s&Eo recomendadas as configura&Ees padr&Eo.
                  O NTFS oferece suporte a 512, 1024, 2048, 4096, 8192,
                  16 K, 32 K, 64 K.
                  O FAT oferece suporte a 512, 1024, 2048, 4096, 8192,
                  16 K, 32 K, 64 K,
                  (128 K, 256 K para setores > 512 bytes).
                  O FAT32 oferece suporte a 512, 1024, 2048, 4096, 8192,
                  16 K, 32 K, 64 K,
                  (128 K, 256 K para setores > 512 bytes).
                  O exFAT oferece suporte a 512, 1024, 2048, 4096, 8192,
                  16 K, 32 K, 64 K,
                  128 K, 256 K, 512 K, 1 M, 2 M, 4 M, 8 M, 16 M, 32 M.

clusters         Os sistemas de arquivos FAT e FAT32 imp&Eam
                  as seguintes restri&Ees em rela&Eo ao n&fnero de
                  em um volume:
FAT: n&fnero de clusters <= 65526
FAT32: 65526 < n&fnero de clusters < 4177918

A formata&Eo ser   imediatamente interrompida se
n&Eo for possivel atender aos requisitos acima usando
o tamanho de cluster especificado.

N&Eo h   suporte para compacta&Eo NTFS em unidades de
aloca&Eo de tamanho superior a 4096.

/F:tamanho      Especifica o tamanho do disquete a ser formatado
(1.44)
```

/T:trilhas Especifica o n mero de trilhas por lado do disco.  
/N:setores Especifica o n mero de setores por trilha.  
/P:count Zerar todos os setores no volume. Depois disso, o  
volume ser substitu do pelo n mero de vezes especificado em  
"count" usando um n mero aleat rio diferente a cada vez. Se  
"count" for igual a zero, nenhuma substitui o adicional  
ser feita ap s serem zerados todos os setores. Essa  
op o, ignorada quando /Q , especificado.  
/S:state Especifica suporte para nomes de arquivo curtos  
(habilitar, desabilitar)  
Nomes curtos s o desabilitados por padr o

/? é um parâmetro inválido.  
---- Comandos compatíveis ----

8dot3name	Gerenciamento de 8dot3name
behavior	Controla o comportamento do sistema de arquivos
dirty	Gerencia bit sujo de volume
file	Comandos específicos de arquivo
fsinfo	Informações do sistema de arquivos
hardlink	Gerenciamento de link físico
objectid	Gerenciamento de IDs de objeto
quota	Gerenciamento de cotas
repair	Gerenciamento de autorrecuperação
reparsepoint	Gerenciamento de ponto de nova análise
resource	Administração do Gerenciador de Recursos de Transação
sparse	Controle de arquivo esparsos
transaction	Gerenciamento de transações
usn	Gerenciamento de USN
volume	Gerenciamento de volume
wim	Gerenciamento de hospedagem do Wim transparente

Exibe ou modifica tipos de arquivos usados nas associações de extensão de arquivo

```
FTYPE [Tipo_de_arquivo[=[Cadeia_de_caracteres_do_comando_open]]]
```

Tipo\_de\_arquivo Especifica o tipo de arquivo a ser examinado ou alterado  
Cadeia\_de\_caracteres\_do\_comando\_open Especifica o comando open  
a ser usado ao iniciar arquivos desse tipo.

Digite FTYPE sem parâmetros para exibir os tipos de arquivos atuais que possuem cadeia de caracteres do comando open definidas. O FTYPE é invocado com apenas um tipo de arquivo; ele exibe a cadeia de caracteres do comando open atual para esse tipo de arquivo. Se nada for especificado para a cadeia de caracteres do comando open, o comando FTYPE excluirá a cadeia de caracteres do comando open para o tipo de arquivo. Dentro de uma cadeia de caracteres do comando open, %0 ou %1 são substituídos pelo nome de arquivo sendo iniciado por associação. %\* obtém todos os parâmetros e %2 obtém o primeiro parâmetro, %3 o segundo, etc. %~n obtém todos os parâmetros restantes iniciados com o enésimo parâmetro, onde

n pode estar entre 2 e 9, inclusive. Por exemplo:

```
ASSOC .pl=PerlScript
FTYPE PerlScript=perl.exe %1 %*
```

permitiria invocar um script Perl como a seguir:

```
script.pl 1 2 3
```

Para eliminar a necessidade de digitar as extensões, faça o seguinte:

```
defina PATHEXT=.pl;%PATHEXT%
```

e o script poderia ser chamado assim:

```
script 1 2 3
```

Direciona o cmd.exe para uma linha com um rótulo em um programa em lotes.

GOTO rótulo

rótulo Especifica a cadeia de caracteres de texto usada no programa  
em  
lotes como um rótulo.

Você deve digitar um rótulo em uma linha iniciada com dois pontos (:).

Se as extensões de comando estiverem ativadas, o GOTO será alterado como  
a seguir:

O comando GOTO agora aceita um rótulo de destino de :EOF que transfere o  
controle para o final do arquivo de script em lotes atual. Essa é uma  
forma

fácil de sair de um arquivo de script em lotes sem definir um rótulo.  
Digite CALL /? para obter uma descrição das extensões para o comando CALL  
que tornam este recurso n útil.

```
GPRESULT [/S sistema [/U nome_usuário [/P [senha]]] [/SCOPE escopo]
          [/USER nome_usuário_destino] [/R | /V | /Z] [(/X | /H)
          <nome_arquivo> [/F]]
```

**Descrição:**

Esta ferramenta de linha de comando exibe as informações do Conjunto de Políticas Resultante (RSOP) para um computador e um usuário de destino.

**Lista de Parâmetros:**

/S	sistema	Especifica o sistema remoto com o qual será estabelecida a conexão.
/U	[domínio\]usuário	Especifica o contexto de usuário sob o qual o comando deve ser executado. Não pode ser usado com /X, /H.
/P	[senha]	Especifica a senha do contexto de usuário fornecido. Solicita uma entrada, se omitido. Não pode ser usado com /X, /H.
/SCOPE	escopo configurações	Especifica se o usuário ou as configurações do computador precisam ser exibidas. Valores válidos: "USER", "COMPUTER".
/USER	[domínio\]usuário	Especifica o nome de usuário para o qual os dados RSOP devem ser exibidos.
/X	<nome de arquivo>	Salva o relatório em formato XML no local e com o nome de arquivo especificado pelo parâmetro <nome de arquivo>. (Válido no Windows Vista SP1 e posterior e Windows Server 2008 e posterior)
/H	<nome de arquivo>	Salva o relatório em formato HTML local e com o nome de arquivo especificado pelo parâmetro <nome de arquivo>. (Válido no Windows no mínimo Vista SP1 e no mínimo Windows Server 2008)
/F	arquivo	Força gpresult a substituir o nome de especificado no comando /X ou /H.
/R		Exibe dados resumidos de RSOP.
/V	fornecem	Especifica a exibição de informações detalhadas. As informações detalhadas configurações adicionais detalhadas que foram aplicadas com precedência 1.

/z	Especifica que as informações superdetalhadas devem ser exibidas. As informações superdetalhadas fornecem configurações adicionais detalhadas que foram aplicadas com precedência 1 e superior. Isso permite ver se uma configuração foi definida em diversos locais. Consulte o tópico da ajuda online sobre Política de Grupo para obter mais informações.
/?	Exibe esta mensagem de ajuda.

## Exemplos:

```
GPRESULT /R  
GPRESULT /H GPReport.html  
GPRESULT /USER nome_usuário_destino /V  
GPRESULT /S sistema /USER nome_usuário_destino /SCOPE COMPUTER /Z  
GPRESULT /S sistema /U nome_usuario /P senha /SCOPE USER /V
```

Fornece informações de ajuda sobre comandos do Windows.

HELP [comando]

comando - exibe informações de ajuda sobre o comando.

ICACLS name /save aclfile [/T] [/C] [/L] [/Q]  
armazena as DACLs referentes aos arquivos e ...s pastas correspondentes  
a name em arquivo\_ACL para uso posterior com /restore. Observe que  
as SACLs, o propriet rio e os r tulos de integridade n o s o salvos.

ICACLS directory [/substitute SidOld SidNew [...]] /restore aclfile  
[/C] [/L] [/Q]  
aplica as DACLs armazenadas aos arquivos de diret rio.

ICACLS name /setowner user [/T] [/C] [/L] [/Q]  
altera o propriet rio de todos os nomes correspondentes. Essa op o  
n o for a uma altera o de propriedade; para essa finalidade, use  
o utilit rio takeown.exe.

ICACLS name /findsid Sid [/T] [/C] [/L] [/Q]  
localiza todos os nomes correspondentes que cont m uma ACL  
com men o expl cita a Sid.

ICACLS name /verify [/T] [/C] [/L] [/Q]  
localiza todos os arquivos cuja ACL n o est na forma can nica  
ou cujo tamanho , inconsistente com as contagens de ACEs.

ICACLS name /reset [/T] [/C] [/L] [/Q]  
substitui as ACLs por ACLs herdadas padr o para todos os arquivos  
correspondentes.

ICACLS name [/grant[:r] Sid:perm[...]]  
[/deny Sid:perm [...]]  
[/remove[:g|:d] Sid[...]] [/T] [/C] [/L] [/Q]  
[/setintegritylevel Level:policy[...]]

/grant[:r] Sid:perm concede direitos de acesso ao usu rio  
especificado.

Com :r, as permiss es substituem todas as permiss es expl citas  
concedidas anteriormente. Sem :r, as permiss es s o adicionadas  
...s permiss es expl citas concedidas anteriormente.

/deny Sid:perm nega explicitamente direitos de acesso ao usu rio  
especificado. Uma ACE de nega o expl cita , adicionada para as  
permiss es declaradas e as mesmas permiss es em concess es  
expl citas  
s o removidas.

/remove[:[g|d]] Sid remove todas as ocorr ncias de SID na ACL. Com  
:g, remove todas as ocorr ncias de direitos concedidos a essa  
SID.

Com :d, remove todas as ocorr ncias de direitos negados a essa  
Sid.

/setintegritylevel [(CI) (OI)]Level adiciona explicitamente uma ACE de  
integridade a todos os arquivos correspondentes. O n vel  
dever ser  
especificado como um destes:

L[ow]  
M[edium]  
H[igh]

Op es de heran a para a ACE de integridade podem preceder o  
n vel  
e s o aplicadas somente a diret rios

```
/inheritance:e|d|r  
e - habilita a herança  
d - desabilita a herança e copia as ACEs  
r - remove todas as ACEs herdadas
```

#### Observações:

As SIDs podem estar no formato num,rico ou de nome amig vel. Se for usado o formato num,rico, afixe um \* ao inicio da SID.

/T indica que a operação ser executada em todos os arquivos/diretórios correspondentes abaixo dos diretórios especificados em nome.

/C indica que a operação continuar em todos os erros de arquivo. As mensagens de erro ainda serão exibidas.

/L indica que a operação ser executada em um link simbólico propriamente dito, e não no destino correspondente.

/Q indica que icacls deve suprimir as mensagens de ^xito.

ICACLS preserva a ordenação canônica das entradas ACE:

Negações explícitas  
Concessões explícitas  
Negações herdadas  
Concessões herdadas

perm , uma máscara de permissão e pode ser especificada de duas formas:

uma sequência de direitos simples:

N - sem acesso  
F - acesso completo  
M - acesso para modificar  
RX - racesso para ler e executar  
R - acesso somente leitura  
W - acesso somente gravação  
D - acesso para excluir

uma lista separada por vírgulas de direitos específicos, entre:

DE - excluir  
RC - ler controle  
WDAC - gravar DAC  
WO - gravar proprietário  
S - sincronizar  
AS - acessar segurança do sistema  
MA - máximo permitido  
GR - leitura genérica  
GW - gravação genérica  
GE - execução genérica  
GA - todos genéricos  
RD - ler dados/listar diretório  
WD - gravar dados/adicionar arquivo  
AD - acrescentar dados/adicionar subdiretório  
REA - ler atributos estendidos  
WEA - gravar atributos estendidos  
X - executar/atravessar  
DC - excluir filho  
RA - ler atributos  
WA - gravar atributos

os direitos de heran a podem preceder qualquer uma das duas formas

e aplicam-se somente a diret rios:

- (OI) - objetos s o herdeiros
- (CI) - cont ineres s o herdeiros
- (IO) - aplica-se somente aos herdeiros
- (NP) - n o propagar heran a
- (I) - permiss o herdada do cont iner pai

Exemplos:

icacls c:\windows\\* /save AclFile /T

- Salva as ACLs referentes a todos os arquivos de c:\windows e seus subdiret rios em AclFile.

icacls c:\windows\ /restore AclFile

- Restaura as ACLs correspondentes a cada arquivo inclu do em AclFile que exista em c:\windows e seus subdiret rios.

icacls file /grant Administrator:(D,WDAC)

- Concede ao usu rio Administrador as permiss es Excluir e Gravar DAC para arquivo.

icacls file /grant \*S-1-1-0:(D,WDAC)

- Concede ao usu rio definido pela sid S-1-1-0 as permiss es Excluir e Gravar DAC para arquivo.

Executa o processamento condicional nos programas em lotes.

IF [NOT] ERRORLEVEL número comando	
IF [NOT] cadeia_de_caracteres1==cadeia_de_caracteres2 comando	
IF [NOT] EXIST nome_de_arquivo comando	
NOT	Especifica que o Windows só deve executar o comando se a condição for falsa.
ERRORLEVEL número	Especifica uma condição verdadeira se o último programa executado retornar um código de saída igual ou maior que o número especificado.
cadeia_de_caracteres1==cadeia_de_caracteres2	Especifica uma condição verdadeira se as cadeias de caracteres de texto especificadas forem correspondentes.
EXIST nome_de_arquivo	Especifica uma condição verdadeira se o nome de arquivo especificado existir.
comando condição	Especifica o comando a ser executado se a condição for atendida. O comando pode ser seguido pelo comando ELSE, que executará o comando após a palavra-chave ELSE se a condição especificada for FALSA

A cláusula ELSE deve ocorrer na mesma linha que o comando após o IF. Por exemplo:

```
IF EXIST nome_de_arquivo. (
    del nome_de_arquivo.
) ELSE (
    echo nome_de_arquivo. ausente.
)
```

O exemplo a seguir NÃO funcionaria porque o comando del precisa ser terminado por uma nova linha:

```
IF EXIST nome_de_arquivo. del nome_de_arquivo. ELSE echo
nome_de_arquivo.
ausente
```

O exemplo a seguir também não funcionaria, já que o comando ELSE deve estar na mesma linha que o final do comando IF:

```
IF EXIST nome_de_arquivo. del nome_de_arquivo.
ELSE echo nome_de_arquivo. ausente
```

O exemplo a seguir funcionaria se você desejasse tudo isso em uma linha:

```
IF EXIST nome_de_arquivo. (del nome_de_arquivo.) ELSE echo
nome_de_arquivo. ausente
```

Se as extensões de comando estiverem ativadas, o IF será alterado como a seguir:

```
IF [/I] cadeia de caracteres1 op_comparação cadeia de caracteres2
comando
IF CMDEXTVERSION número comando
IF DEFINED variável comando
```

onde op\_comparação pode ser uma das seguintes:

```
EQU - igual
NEQ - diferente
LSS - menor que
LEQ - menor que ou igual
GTR - maior que
GEQ - maior que ou igual
```

e a opção /I, se especificada, informa para fazer comparações de cadeias de caracteres sem diferenciação de maiúsculas e minúsculas. A opção /I também pode ser usada na fórmula cadeia de caracteres1==cadeia de caracteres2 de IF. Essas comparações são genéricas, pois se cadeia de caracteres1 e cadeia de caracteres2 contiverem todos os dígitos numéricos, as cadeias de caracteres serão convertidas em números e será executada uma comparação numérica.

A condicional CMDEXTVERSION funciona como ERRORLEVEL, exceto por comparar com um número de versão interno associado às extensões de comandos. A primeira versão é 1. Ela será incrementada em um quando melhorias significativas forem adicionadas às extensões de comandos. A condicional CMDEXTVERSION nunca é verdadeira quando as extensões de comandos estão desativadas.

A condicional DEFINED funciona como EXISTS, exceto por usar um nome de variável de ambiente e ser verdadeira se a variável de ambiente estiver definida.

%ERRORLEVEL% expandirá para uma representação de cadeia de caracteres do valor atual de ERRORLEVEL, contanto que ainda não exista uma variável ambiente com o mesmo nome ERRORLEVEL, pois; nesse caso, será obtido o valor. Após executar um programa, o exemplo a seguir ilustra o uso ERRORLEVEL:

```
goto resposta%ERRORLEVEL%
:resposta0
echo O programa retornou o código 0
:resposta1
echo O programa retornou o código 1.
```

Também é possível usar as comparações numéricas acima:

```
IF %ERRORLEVEL% LEQ 1 goto okay
```

%CMDCMDLINE% expandirá para a linha de comando original transferida para CMD.EXE antes de qualquer processamento pelo CMD.EXE, contanto que ainda não haja uma variável de ambiente com o nome CMDCMDLINE, pois, nesse caso, será o seu valor.

%CMDEXTVERSION% expandirá para uma representação da cadeia de caracteres do valor atual de CMDEXTVERSION, contanto que ainda não exista uma variável de ambiente com o nome CMDEXTVERSION, pois, neste caso, será obtido seu valor.

Cria, altera ou exclui o rótulo de volume de um disco.

LABEL [unidade:] [rótulo]  
LABEL [/MP] [volume] [rótulo]

unidade:	Especifica a letra de uma unidade.
rótulo	Especifica o rótulo do volume.
/MP	Especifica que o volume deve ser tratado como um ponto de montagem ou nome de volume.
volume	Especifica a letra de unidade (seguida de dois-pontos), o ponto de montagem ou o nome de volume. Se o nome de volume
	for especificado, o sinalizador /MP ser desnecessário.

Cria uma pasta.

```
MKDIR [unidade:]caminho  
MD [unidade:]caminho
```

Se as extensões de comando estiverem ativadas, MKDIR será alterado como a seguir:

MKDIR cria quaisquer pastas intermediárias no caminho, se necessário. Por exemplo, se \a não existisse:

```
mkdir \a\b\c\d
```

seria o mesmo que:

```
mkdir \a  
chdir \a  
mkdir b  
chdir b  
mkdir c  
chdir c  
mkdir d
```

que seria o que você teria de digitar se as extensões estivessem desativadas.

Cria uma pasta.

```
MKDIR [unidade:]caminho  
MD [unidade:]caminho
```

Se as extensões de comando estiverem ativadas, MKDIR será alterado como a seguir:

MKDIR cria quaisquer pastas intermediárias no caminho, se necessário. Por exemplo, se \a não existisse:

```
mkdir \a\b\c\d
```

seria o mesmo que:

```
mkdir \a  
chdir \a  
mkdir b  
chdir b  
mkdir c  
chdir c  
mkdir d
```

que seria o que você teria de digitar se as extensões estivessem desativadas.

Cria um link simbólico.

MKLINK [[/D] | [/H] | [/J]] Destino do link

/D Cria um link simbólico de diretório. Padrão é um link simbólico de arquivo.

/H Cria um link real em vez de um link simbólico.

/J Cria uma Junção de diretório.

Link especifica o nome do novo link simbólico.

Destino especifica o caminho (relativo ou absoluto) ao qual o novo link se refere.

Configura os dispositivos do sistema.

Porta serial:	MODE COMm[:] [BAUD=b] [PARITY=p] [DATA=d] [STOP=s] [to=on off] [xon=on off] [odsr=on off] [octs=on off] [dtr=on off hs] [rts=on off hs tg] [idsr=on off]
Status do dispositivo:	MODE [dispositivo] [/STATUS]
Redirecionar impressão:	MODE LPTn[:]=COMm[:]
Selecionar página de código:	MODE CON[:] CP SELECT=yyy
Status da página de código:	MODE CON[:] CP [/STATUS]
Modo de exibição:	MODE CON[:] [COLS=c] [LINES=n]
Velocidade de repetição:	MODE CON[:] [RATE=r DELAY=d]

Exibe as informações tela a tela.

MORE [/E [/C] [/P] [/S] [/Tn] [+n]] < [unidade:] [caminho] arquivo  
comando | MORE [/E [/C] [/P] [/S] [/Tn] [+n]]  
MORE /E [/C] [/P] [/S] [/Tn] [+n] [arquivos]

[unidade:] [caminho] arquivo Especifica um arquivo para exibir  
uma tela de cada vez.

comando Especifica o comando que será  
exibido.

/E Habilita recursos estendidos  
/C Limpa a tela antes de exibir a página  
/P Expande os caracteres de avanço de página  
/S Compacta múltiplas linhas em branco em uma única linha  
/Tn Expande as tabulações para n espaços (padrão: 8)

Opções podem ser usadas na variável de ambiente  
MORE.

+n Começa exibindo o primeiro arquivo na linha n

arquivos Lista de arquivos a serem exibidos. Os arquivos  
da lista estão separados por espaços.

Se os recursos estendidos estiverem habilitados, os seguintes  
comandos serão aceitos no prompt -- More --:

P n	Exibir as próximas n linhas
S n	Ignorar as próximas n linhas
F	Exibir o próximo arquivo
Q	Sair
=	Exibir o número de linha
?	Exibir a linha de ajuda
<espaço>	Exibir a próxima página
<enter>	Exibir a próxima linha

Move e renomeia arquivos e pastas.

Para mover um ou mais arquivos:

MOVE [/Y | /-Y] [unidade:] [caminho]arquivo1[,... ] destino

Para renomear uma pasta:

MOVE [/Y | /-Y] [unidade:] [caminho]pasta1 pasta2

[unidade:]	[caminho]arquivo1	Especifica o local e o nome do arquivo ou arquivos a serem movidos.
destino		Especifica o novo local do arquivo. O destino pode consistir em uma letra de unidade e dois-pontos, um nome de pasta ou uma combinação. Se estiver movendo apenas um arquivo, também poderá incluir um nome de arquivo se desejar renomear o arquivo ao movê-lo.
[unidade:]	[caminho]pasta1	Especifica a pasta a ser renomeada.
pasta2		Especifica o novo nome da pasta.
/Y		Suprime o prompt para você confirmar se deseja substituir um arquivo de destino existente.
/-Y		Exibe o prompt para você confirmar se deseja substituir um arquivo de destino existente.

A opção /Y pode estar presente na variável de ambiente COPYCMD.

Isso pode ser substituído por /-Y na linha de comando. O padrão é exibir um prompt nas substituições a menos que o comando MOVE esteja sendo executado em um script em lote.

**OPENFILES /parâmetro [argumentos]**

**Descrição:**

Permite que um administrador liste ou desconecte arquivos e pastas que foram abertos em um sistema.

**Lista de parâmetros:**

/Disconnect	Desconecta um ou mais arquivos abertos.
/Query	Exibe arquivos abertos localmente ou por meio de pastas compartilhadas.
/Local	Ativa/desativa a exibição de arquivos locais abertos.
/?	Exibe esta mensagem de ajuda.

**Exemplos:**

OPENFILES /Disconnect /?  
OPENFILES /Query /?  
OPENFILES /Local /?

Exibe ou define um caminho de pesquisa para arquivos executáveis.

```
PATH [[unidade:]caminho[;...][;%PATH%]  
PATH ;
```

Digite PATH ; para limpar todas as configurações de caminhos de pesquisa e

instruir cmd.exe a pesquisar apenas na pasta atual.

Digite PATH sem parâmetros para exibir o caminho atual.

A inclusão de %PATH% na nova configuração de caminho faz com que o caminho antigo seja acrescentado à nova configuração.

Pausa o processamento de um programa em lotes e exibe a mensagem  
Pressione qualquer tecla para continuar. . .

Altera para a pasta armazenada pelo comando PUSHD.

POPD

Se as extensões de comando estiverem ativadas, o comando POPD excluirá qualquer letra de unidade temporária criada pelo PUSHD quando você usar o POPD para remover essa unidade da pilha de diretórios.

Imprime um arquivo de texto.

PRINT [/D:dispositivo] [[unidade:] [caminho]arquivo[...]]

/D:dispositivo Especifica um dispositivo de impressão.

Altera o prompt de comando do cmd.exe.

PROMPT [texto]

texto Especifica um novo prompt de comando.

O prompt de comando pode constituir-se de caracteres normais e dos seguintes códigos especiais:

\$A	& ("E" comercial)
\$B	(pipe)
\$C	( (parêntese esquerdo)
\$D	data atual
\$E	código de escape (código ASCII 27)
\$F	) (parêntese direito)
\$G	> (sinal de maior que)
\$H	BACKSPACE (elimina o caractere anterior)
\$L	< (sinal de menor que)
\$N	unidade atual
\$P	unidade e caminho atual
\$Q	= (sinal de igual)
\$S	(espaço em branco)
\$T	hora atual
\$V	versão do Windows
\$	retorno de carro e avanço de linha
\$\$	\$ (cifrão)

Se as extensões de comando estiverem ativadas, o comando PROMPT dará suporte aos seguintes caracteres de formatação adicionais:

\$+ zero ou mais caracteres de sinal de mais (+) dependendo da profundidade da pilha de diretórios do comando PUSHD, um caractere para cada nível adicionado.

\$M Exibe o nome remoto associado à letra da unidade atual ou à cadeia de caracteres vazia se a unidade atual não for uma unidade de rede.

Armazena a pasta atual para uso pelo comando POPD, depois altera para a pasta especificada.

PUSHD [caminho | ..]

caminho Especifica a pasta que se tornará a atual.

Se as extensões de comando estiverem ativadas, o comando PUSHD aceitará caminhos de rede além da letra da unidade e do caminho normais.

Se um caminho de rede for especificado, o PUSHD criará uma letra de unidade

temporária que aponte para esse recurso de rede especificado e alterará a unidade e pasta atuais, usando a letra da unidade recém definida. As letras de unidades temporárias são alocadas de Z: para baixo, usando a primeira letra de unidade não usada encontrada.

Remove (exclui) uma pasta.

RMDIR [/S] [/Q] [unidade:]caminho  
RD [/S] [/Q] [unidade:]caminho

- /S Remove todas as pastas e arquivos da pasta especificada, além dela mesma. Utilizado para remover uma árvore de pastas.
- de /Q Modo silencioso. Não pede confirmação para remover a árvore de pastas ao se passar o parâmetro /S.

Recupera as informações legíveis de um disco danificado ou defeituoso.

RECOVER [unidade:] [caminho]arquivo

Consulte a referência aos comandos online na Ajuda do Windows  
antes de usar o comando RECOVER.

Registra comentários em um arquivo em lotes ou no CONFIG.SYS.

REM [comentário]

Renomeia um ou mais arquivos.

```
RENAME [unidade:] [caminho]arquivo1 arquivo2.  
REN [unidade:] [caminho]arquivo1 arquivo2.
```

Note que você não pode especificar uma nova unidade ou caminho para o arquivo de destino.

Renomeia um ou mais arquivos.

```
RENAME [unidade:] [caminho]arquivo1 arquivo2.  
REN [unidade:] [caminho]arquivo1 arquivo2.
```

Note que você não pode especificar uma nova unidade ou caminho para o arquivo de destino.

Substitui arquivos.

REPLACE [unidade1:] [cam1]arquivo [unidade2:] [cam2] [/A] [/P] [/R] [/W]  
REPLACE [unidade1:] [cam1]arquivo [unidade2:] [cam2] [/P] [/R] [/S] [/W]  
[/U]

[unidade1:] [cam1]arquivo Especifica o(s) arquivo(s) de origem.  
[unidade2:] [cam2] Especifica a pasta onde os arquivos deverão ser substituídos.  
/A Adiciona novos arquivos ... pasta de destino.  
NÃO podem ser usados com as opções /S ou /U.  
/P Pede confirmação antes de substituir um arquivo  
ou de adicionar um arquivo de origem.  
/R Substitui arquivos de somente leitura e  
arquivos sem proteção.  
/S Substitui arquivos em todas as subpastas da pasta de destino. NÃO pode ser usado com a opção /A.  
/W Espera um disco ser inserido antes de começar.  
/U Substitui (atualiza) somente arquivos mais  
que os arquivos de origem. NÃO pode ser usado com a opção /A.

Remove (exclui) uma pasta.

RMDIR [/S] [/Q] [unidade:]caminho  
RD [/S] [/Q] [unidade:]caminho

- /S Remove todas as pastas e arquivos da pasta especificada, além dela mesma. Utilizado para remover uma árvore de pastas.
- de /Q Modo silencioso. Não pede confirmação para remover a árvore de pastas ao se passar o parâmetro /S.

```
-----  
-----  
ROBOCOPY      ::      Robust File Copy para Windows  
-----  
-----  
  
Iniciado: quinta-feira, 14 de março de 2019 03:10:10 pm  
Uso :: ROBOCOPY origem destino [arquivo  
[arquivo]...]  
          [opções]  
  
origem :: Diretório de Origem (unidade:\caminho ou  
          \\servidor\compartilhamento\caminho).  
destino :: Diretório de Destino (unidade:\caminho ou  
          \\servidor\compartilhamento\caminho).  
arquivo :: Arquivo(s) a serem copiados  
          (nomes/curingas: o padrão é "*.*").  
  
::  
:: Opções de cópia :  
::  
          /S :: copiar subdiretórios, mas não os vazios.  
          /E :: copiar subdiretórios, incluindo os vazios.  
          /LEV:n :: copiar somente os níveis superiores da  
árvore  
          de diretórios de origem.  
  
          /Z :: copiar arquivos no modo reiniciável.  
          /B :: copiar arquivos no modo de Backup.  
          /ZB :: usar o modo reiniciável; se o acesso for  
negado,  
          use o modo de Backup.  
          /J :: copiar usando E/S não armazenada em buffer  
          (recomendável para arquivos grandes).  
          /EFSRAW :: copiar todos os arquivos criptografados no  
modo  
          EFS RAW.  
  
          /COPY:marca[s] :: o que COPIAR em arquivos (o padrão é  
/COPY:DAT).  
          (marcas : D=Dados, A=Atributos, T=Carimbos  
de  
          Data/Hora).  
          (S=Segurança=ACLs NTFS, O=Informações do  
proprietário, U=Informações de auditoria).  
  
          /SEC :: copiar arquivos com segurança (equivalente a  
          /COPY:DATS).  
          /COPYALL :: COPIAR TODAS as informações do arquivo  
          (equivalente a /COPY:DATSOU).  
          /NOCOPY :: COPIAR NENHUMA informação do arquivo (útil  
com  
          /PURGE).  
          /SECFIX :: CORRIGIR a segurança de arquivo em todos os  
          arquivos, mesmo em arquivos ignorados.  
          /TIMFIX :: CORRIGIR as horas do arquivo em todos os  
          arquivos, mesmo nos arquivos ignorados.
```

/PURGE :: excluir arquivos/diretórios de destino que  
não existam mais na origem.  
/MIR :: espelhar uma árvore de diretórios  
(equivalente a /E mais /PURGE).  
  
/MOV :: mover arquivos (excluir da origem após  
copiar).  
/MOVE :: MOVER arquivos E diretórios (excluir da  
origem após copiar).  
  
/A+:[RASHCNET] :: adicionar os atributos fornecidos aos  
arquivos copiados.  
/A-:[RASHCNET] :: remover os atributos fornecidos aos arquivos  
copiados.  
  
/CREATE :: CRIAR árvore de diretórios e arquivos de  
comprimento nulo somente.  
/FAT :: criar arquivos de destino usando somente  
nomes de arquivo FAT 8.3.  
/256 :: desativar suporte a caminho muito  
longo (> 256 caracteres).  
  
/MON:n :: MONitorar origem; executar novamente quando  
mais de n alterações forem vistas.  
/MOT:m :: monitorar origem; executar novamente em m  
minutos, se alterado.  
  
/RH:hhmm-hhmm :: Hora de Execução - horário em que novas  
cópias podem ser iniciadas.  
/PF :: verificar horas de execução por  
arquivo (não por transmissão).  
  
/IPG:n :: Intervalo entre Pacotes (ms), para liberar  
largura de banda em linhas lentas.  
  
/SL :: copiar links simbólicos versus o destino.  
  
/MT[:n] :: fazer cópias com vários threads, com n  
threads (padrão 8).  
n deve ser pelo menos 1 e não ser superior a  
128.  
Esta opção é incompatível com as opções /IPG  
e /EFSRAW.  
Redirecione a saída usando a opção /LOG para  
obter um melhor desempenho.  
  
/DCOPY:marca[s] :: o que COPIAR para diretórios (padrão é  
/DCOPY:DA).  
(marcas : D=Dados, A=Atributos, T=Carimbos  
de Data/Hora).  
  
/NODCOPY :: COPIAR NENHUMA informação do diretório (por  
padrão /DCOPY:DA está concluído).

```
/NOOFFLOAD :: copia arquivos sem usar o mecanismo  
             Descarregamento da Cópia do Windows.  
  
::  
:: Opções de Seleção de Arquivo :  
::  
    /A :: copiar somente os arquivos com o conjunto de  
          atributos de Arquivamento.  
    /M :: copiar somente os arquivos com o atributo de  
          Arquivamento e redefini-los.  
    /IA:[RASHCNETO] :: Incluir somente arquivos com qualquer um dos  
                      conjuntos determinados de Atributos.  
    /XA:[RASHCNETO] :: eXcluir arquivos com qualquer um dos  
                      conjuntos  
                           determinados de Atributos.  
  
    /XF arquivo [arquivo]... :: eXcluir arquivos que correspondem a  
                           determinados  
                           nomes/caminhos/curingas.  
    /XD dirs [dirs]... :: eXclude Diretórios que correspondem a  
                           determinados  
                           nomes/caminhos.  
  
    /XC :: eXcluir arquivos alterados.  
    /XN :: eXcluir arquivos mais recentes.  
    /XO :: eXcluir arquivos mais antigos.  
    /XX :: eXcluir arquivos e diretórios eXtra.  
    /XL :: eXcluir arquivos e diretórios solitários.  
    /IS :: Incluir os mesmos arquivos.  
    /IT :: Incluir arquivos ajustados.  
  
    /MAX:n :: tamanho MÁXimo de arquivo - excluir arquivos  
              maiores que n bytes.  
    /MIN:n :: tamanho MÍNimo de arquivo - excluir arquivos  
              menores que n bytes.  
  
    /MAXAGE:n :: idade de arquivo MÁXima - excluir arquivos  
mais  
              antigos que n dias/data.  
    /MINAGE:n :: idade de arquivo MÍNima - excluir arquivos  
mais  
              recentes que n dias/data.  
    /MAXLAD:n :: última data de acesso máxima - excluir  
arquivos  
              não utilizados desde n.  
    /MINLAD:n :: última data de acesso mínima - excluir  
arquivos  
              utilizados desde n.  
              (Se n < 1900, então n = n dias ou n = data  
               AAAAMMDD) .  
  
    /XJ :: eXcluir pontos de Junção. (normalmente  
incluído  
              por padrão).  
  
    /FFT :: assumir horas de arquivo FAT (granularidade  
           de 2 segundos).  
    /DST :: compensar para diferenças de uma hora de  
           horário de verão.
```

```
/XJD :: eXcluir pontos de Junção para Diretórios.  
/XJF :: eXcluir pontos de Junção para arquivos.  
  
::  
:: Opções de Repetição :  
::  
/R:n :: número de Repetições em cópias com falhas:  
        o padrão é 1 milhão.  
/W:n :: tempo de espera entre as repetições: o  
padrão  
        é 30 segundos.  
  
/REG :: Salve /R:n e /W:n no Registro como  
configurações  
        padrão.  
  
/TBD :: aguardar nomes de compartilhamentos a serem  
definidos (erro de repetição 67).  
  
::  
:: Opções de Log :  
::  
/L :: Listar somente - não copiar, usar carimbo de  
        data/hora ou excluir qualquer arquivo.  
/X :: relatar todos os arquivos extra, não apenas  
os  
        selecionados.  
/V :: produzir saída detalhada, mostrando arquivos  
        ignorados.  
/TS :: incluir carimbo de data/hora no arquivo de  
        origem na saída.  
/FP :: incluir nome de caminho completo de arquivos  
na  
        saída.  
/BYTES :: Imprimir tamanhos como bytes.  
  
/NS :: sem tamanho - não registrar tamanhos de  
arquivo.  
/NC :: sem classe - não registrar classes de  
arquivo.  
/NFL :: sem lista de arquivos - não registrar nomes  
de  
        arquivo.  
/NDL :: sem lista de diretórios - não registrar  
nomes  
        de diretório.  
  
/NP :: sem progresso - não exibir percentual  
copiado.  
/ETA :: mostrar tempo estimado de chegada dos  
arquivos  
        copiados.  
  
/LOG:arquivo :: status de saída para arquivos de log  
        (substituir log existente).  
/LOG+:arquivo :: status de saída para arquivos de log  
        (anexar a log existente).  
  
/UNILOG:arquivo :: status de saída para arquivos de log como  
UNICODE
```

```
          (substituir log existente).
/UNILOG+:arquivo :: status de saída para arquivos de log como
UNICODE
                           (anexar a log existente).

arquivo
       /TEE :: saída para janela de console, assim como
              de log.

/NJH :: sem descrição de trabalho.
/NJS :: sem resumo de trabalho.

/UNICODE :: status de saída como UNICODE.

:::
::: Opções de Trabalho :
:::
/JOB:trabalho :: pegar parâmetros do arquivo de trabalho
nomeado.
/SAVE:trabalho :: salvar parâmetros no arquivo de trabalho
nomeado
      /QUIT :: sair depois de processar a linha de
              comando (para exibir parâmetros).
      /NOSD :: nenhum diretório de origem especificado.
      /NODD :: nenhum diretório de destino especificado.
      /IF :: incluir os seguintes arquivos.

:::
::: Comentários :
:::
O uso de /PURGE ou /MIR no diretório raiz do volume
fará robocopy aplicar a operação solicitada também nos arquivos do
diretório Informações do Volume do Sistema. Se isso não for
desejado, a opção /XD poderá ser usada como instrução para
robocopy
      ignorar esse diretório.
```

```
SCHTASKS /parameter [argumentos]
```

**Descrição:**

Permite que um administrador crie, exclua, consulte, altere, execute e termine tarefas agendadas em um sistema local ou remoto.

**Lista de parâmetros:**

/Create	Cria uma nova tarefa agendada.
/Delete	Exclui a(s) tarefa(s) agendada(s).
/Query	Exibe toda(s) a(s) tarefa(s) agendada(s).
/Change	Altera as propriedades de uma tarefa agendada.
/Run	Executa a tarefa agendada por demanda.
/End	Interrompe a tarefa agendada que está em execução no momento.
/ShowSid	Mostra o identificador de segurança que corresponde a um nome de tarefa agendada.
/?	Exibe esta mensagem de ajuda.

**Examples:**

```
SCHTASKS  
SCHTASKS /?  
SCHTASKS /Run /?  
SCHTASKS /End /?  
SCHTASKS /Create /?  
SCHTASKS /Delete /?  
SCHTASKS /Query /?  
SCHTASKS /Change /?  
SCHTASKS /ShowSid /?
```

Exibe, define ou remove variáveis de ambiente do cmd.exe.

SET [variável=[cadeia\_de\_caracteres]]

variável	Especifica o nome da variável de ambiente.
cadeia_de_caracteres	Especifica uma série de caracteres a serem atribuídos à variável.

Digite SET sem parâmetros para exibir as variáveis de ambiente atuais.

Se as extensões de comando estiverem ativadas, SET será alterado como a seguir:

O comando SET chamado com apenas um nome de variável, nenhum sinal de igual ou valor exibirá o valor de todas as variáveis cujo prefixo corresponda ao nome fornecido ao comando SET. Por exemplo:

SET P

exibiria todas as variáveis começando com a letra 'P'

O comando SET definirá o ERRORLEVEL como 1 se o nome da variável não for encontrado no ambiente atual.

O comando SET não permitirá que um sinal de igual seja parte do nome de uma variável.

Duas novas opções foram adicionadas ao comando SET:

SET /A expressão  
SET /P variável=[cadeia\_do\_prompt]

A opção /A especifica que a cadeia de caracteres à direita do sinal de igual é uma expressão numérica que é avaliada. O avaliador da expressão é muito simples e dá suporte às seguintes operações, em ordem decrescente de precedência:

( )	- agrupamento
! ~ -	- operadores unários
* / %	- operadores aritméticos
+ -	- operadores aritméticos
<< >>	- alternância lógica
&	- bit a bit E
^	- bit a bit exclusivo OU
	- bit a bit OU
= *= /= %= += -=	- atribuição
&= ^=  = <<= >>=	
,	- separador de expressões

Se você usar qualquer um dos operadores lógicos ou de módulo, precisará colocar a cadeia de caracteres da expressão entre aspas. Qualquer cadeia de caracteres não numérica na expressão é tratada como nomes de variável de ambiente cujos valores são convertidos para números antes de serem usados. Se um nome de variável de ambiente for especificado, mas não estiver definido no ambiente atual, será usado um valor de zero.

Isso permite fazer aritmética com valores de variáveis de ambiente, sem ter de digitar todos esses sinais de % para obter os valores. Se SET /A for executado a partir da linha de comando fora de um script de comando, ele exibirá o valor final da expressão. O operador de atribuição requer um nome de variável de ambiente à sua esquerda. Os valores numéricos são valores decimais, a menos que sejam antecedidos por 0x para números hexadecimais e 0 para números octais. Portanto, 0x12 é o mesmo que 18 e o mesmo que 022. Observe que a notação octal pode causar confusão: 08 e 09 não são números válidos porque 8 e 9 não são dígitos octais válidos.

A opção /P permite definir o valor de uma variável para uma linha de entrada digitada pelo usuário. Exibe a cadeia de caracteres do prompt especificada antes de ler a linha de entrada. A cadeia de caracteres do prompt pode estar vazia.

A substituição da variável de ambiente foi aprimorada da seguinte forma:

%PATH:seq1=seq2%

expandiria a variável de ambiente PATH, substituindo cada ocorrência de "seq1" no resultado expandido por "seq2". "Seq2" pode ser a cadeia de caracteres vazia para excluir efetivamente todas as ocorrências de "seq1" da saída expandida. "seq1" pode começar com um asterisco e, neste caso, corresponderia a tudo desde o início da saída expandida até a primeira ocorrência da parte restante de seq1.

Também pode especificar subcadeias de caracteres para uma expansão.

%PATH:~10,5%

expandiria a variável de ambiente PATH e usaria apenas os 5 caracteres que começasse no caractere 11 (deslocamento 10) do resultado expandido. Se o comprimento não for especificado, será padronizado como o restante do valor da variável. Se qualquer um dos números (deslocamento ou comprimento) for negativo, o número usado será o comprimento do valor da variável de ambiente adicionado ao deslocamento ou comprimento especificado.

%PATH:~-10%

extrairia os últimos 10 da variável PATH.

%PATH:~0,-2%

extrairia todos os caracteres da variável PATH, com exceção dos 2 últimos.

Finalmente, foi adicionado o suporte à expansão de variáveis de ambiente atrasada. Esse suporte está sempre desativado por padrão, mas pode ser ativado/desativado através da opção da linha de comando /V do CMD.EXE.

Consulte CMD /?

A expansão de variáveis de ambiente atrasada é útil para contornar as limitações da expansão atual que ocorre quando uma linha de texto é lida, e não quando é executada. O exemplo a seguir demonstra o problema com a expansão de variável imediata:

```
set VAR=antes
if "%VAR%" == "antes" (
    set VAR=depois
    if "%VAR%" == "depois" @echo Se você ler isto, terá funcionado
)
```

nunca exibiria a mensagem, já que %VAR% em AMBAS as instruções IF é substituído quando a primeira instrução IF é lida, pois inclui logicamente o corpo do IF, que é uma instrução composta. Portanto, o IF dentro da instrução composta está realmente comparando "antes" com "depois", que nunca será igual. Da mesma forma, o exemplo a seguir não funcionará como esperado:

```
set LIST=
for %i in (*) do set LIST=%LIST% %i
echo %LIST%
```

porque NÃO criará uma lista de arquivos na pasta atual mas, em vez disso, apenas definirá a variável LIST como o último arquivo encontrado.

Novamente, isso ocorre porque a %LIST% é expandida apenas uma vez quando a instrução FOR é lida, e nesse momento a variável LIST está vazia.

Portanto, o loop FOR que está de fato sendo executado é:

```
for %i in (*) do set LIST= %i
```

que apenas continua definindo LIST como o último arquivo encontrado.

A expansão de variáveis de ambiente atrasada permite usar um caractere diferente (o ponto de exclamação) para expandir variáveis de ambiente no tempo de execução. Se a expansão de variáveis atrasada estiver ativada, os exemplos acima poderão ser escritos da seguinte forma para funcionar como o desejado:

```
set VAR=antes
if "%VAR%" == "antes" (
    set VAR=depois
    if "!VAR!" == "depois" @echo Se você ler isto, terá funcionado
)

set LIST=
for %i in (*) do set LIST=!LIST! %i
echo %LIST%
```

Se as extensões de comando estiverem ativadas, haverá diversas variáveis de ambiente dinâmicas que poderão ser expandidas, mas que não aparecerão na lista de variáveis exibida pelo SET. Esses valores de variáveis são computados dinamicamente sempre que o valor da variável é expandido. Se o usuário definir explicitamente uma variável com um desses nomes, essa definição substituirá a definição dinâmica descrita abaixo:

%CD% - expande para a cadeia de caracteres da pasta atual.

%DATE% - expande para a data atual usando o mesmo formato que o comando DATE.

%TIME% - expande para a hora atual usando o mesmo formato que o comando TIME.

%RANDOM% - expande para um número decimal aleatório entre 0 e 32767.

%ERRORLEVEL% - expande para o valor ERRORLEVEL atual

%CMDEXTVERSION% - expande para o número da versão das extensões do processador de comandos atual.

%CMDCMDLINE% - expande para a linha de comando original que chamou o processador de comandos.

%HIGHESTNUMANODENUMBER% - expande para o número de nó NUMA mais alto nesta máquina.

Inicia a localização das alterações de ambiente em um arquivo em lotes. As alterações de ambiente feitas após SETLOCAL ser emitido são específicas

do arquivo em lotes. ENDLOCAL deve ser emitido para restaurar as configurações anteriores. Quando o final de um script em lotes é alcançado, um ENDLOCAL é executado para qualquer comando SETLOCAL pendente emitido por esse script em lotes.

#### SETLOCAL

Se as extensões de comando estiverem habilitadas, o SETLOCAL será alterado como a seguir:

O comando em lotes SETLOCAL agora aceita argumentos opcionais:

ENABLEEXTENSIONS / DISABLEEXTENSIONS  
habilita ou desabilita as extensões do processador de comandos.

Esses argumentos têm precedência sobre as opções CMD /E:ON or /E:OFF Consulte CMD /? para obter detalhes.

ENABLEDELAYEDEXPANSION / DISABLEDELAYEDEXPANSION  
habilita ou desabilita a expansão da variável de ambiente atrasada. Esses argumentos têm precedência sobre as opções

#### CMD

/V:ON or /V:OFF. Consulte CMD /? para obter detalhes.

Essas modificações duram até o comando ENDLOCAL correspondente, independentemente da sua configuração anterior ao comando SETLOCAL.

O comando SETLOCAL definirá o valor ERRORLEVEL se um argumento for fornecido. Ele será zero se um dos dois argumentos válidos for fornecido; caso contrário, será um. É possível usar esses scripts em lotes para determinar se as extensões estão disponíveis usando a seguinte técnica:

```
VERIFY OTHER 2>nulo
SETLOCAL ENABLEEXTENSIONS
IF ERRORLEVEL 1 echo Não é possível habilitar as extensões
```

Isso funciona porque nas versões mais antigas do CMD.EXE, SETLOCAL NÃO define o valor ERRORLEVEL. O comando VERIFY com um argumento incorreto inicializa o valor ERRORLEVEL para um valor diferente de zero.

Altera a posição dos parâmetros substituíveis em um arquivo em lotes.

SHIFT [/n]

Se as extensões de comando estiverem ativadas, o comando SHIFT oferecerá suporte à opção /n que informa ao comando para iniciar a alternância no enésimo argumento, onde n pode estar entre zero e oito. Por exemplo:

SHIFT /2

alternaria de %3 para %2, de %4 para %3, etc. e deixaria %0 e %1 inalterados.

Uso: SHUTDOWN [/i | /l | /s | /r | /g | /a | /p | /h | /e | /o] [/hybrid]  
[/f]  
[/m \\computador] [/t xxx] [/d [p:]xx:yy [/c "comentário"]]

Sem args            Exibir ajuda. É o mesmo que digitar /?.

/?                Exibir ajuda. É o mesmo que não digitar nenhuma opção.

/i                Exibir a interface gráfica do usuário (GUI).

                  Esta deve ser a primeira opção.

/l                Fazer logoff. Não deve ser usada com a opção /m ou /d

/s                Desligar o computador.

/r                Desligamento completo e reinicialização do computador.

/g                Desligamento completo e reinicialização do computador.

Depois que o sistema for  
reiniciado, reinicie todos os aplicativos registrados.

/a                Anular um desligamento do sistema.

                  Só pode ser usado durante o período de tempo limite.

/p                Desligar o computador local sem nenhum tempo limite ou  
aviso.

                  Pode ser usado com as opções /d e /f.

/h                Hibernar o computador local.

                  Pode ser usado com a opção /f.

/hybrid          Executa um desligamento do computador e o prepara para  
inicialização rápida.

                  Deve ser usado com a opção /s.

/e                Documentar o motivo do encerramento inesperado de um  
computador.

/o                Vá para o menu avançado de opções de reinicialização e  
reinic peace o computador.

                  Deve ser usado com a opção /r.

/m \\computer Especificar o computador de destino.

/t xxx          Definir o período de tempo limite antes do desligamento  
como xxx segundos.

                  O intervalo válido é de 0 a 315360000 (10 anos), com um  
padrão de 30.

                  Se o tempo limite for maior que 0, o parâmetro /f será  
implícito.

/c "comment" Comentar o motivo da reinicialização ou do desligamento.  
                  Máximo de 512 caracteres permitidos.

/f                Forçar o fechamento de aplicativos em execução sem avisar  
os usuários.

                  O parâmetro /f é implícito quando um valor maior que 0 é  
especificado para o parâmetro /t.

/d [p|u:]xx:yy Forneça o motivo da reinicialização ou do  
desligamento.

                  p indica que a reinicialização ou o desligamento é  
planejado.

                  u indica que o motivo é definido pelo usuário.

                  Se nem p nem u forem especificados, o reinício ou  
desligamento não será  
planejado.

                  xx é o número do motivo primário (inteiro positivo menor  
que 256).

                  yy é o número do motivo secundário (inteiro positivo menor  
que 65536).

Razões neste computador:  
(E = Esperado U = Inesperado P = planejado, C = definido pelo cliente)

Tipo	Primária	Secundária	Título
U	0	0	Outro (não planejada)

E	0	0	Outro (não planejada)
E P	0	0	Outro (planejado)
U	0	5	Outra falha: o sistema não está respondendo
E	1	1	Hardware: manutenção (não planejada)
E P	1	1	Hardware: manutenção (planejada)
E	1	2	Hardware: instalação (não planejada)
E P	1	2	Hardware: instalação (planejada)
E	2	2	Sistema Operacional: Recuperação (não planejado)
E P	2	2	Sistema Operacional: Recuperação (planejada)
P	2	3	Sistema operacional: atualização (planejada)
E	2	4	Sistema operacional: reconfig. (não planejada)
E P	2	4	Sistema operacional: reconfig. (planejada)
P	2	16	Sistema operacional: service pack (planejado)
	2	17	Sistema operacional: hotfix (não planejado)
P	2	17	Sistema operacional: hotfix (planejado)
	2	18	Sistema operacional: correção de segurança (não planejado)
P	2	18	Sistema operacional: correção de segurança (planejada)
E	4	1	Aplicativo: manutenção (não planejada)
E P	4	1	Aplicativo: manutenção (planejada)
E P	4	2	Aplicativo: instalação (planejada)
E	4	5	Aplicativo: sem resposta
E	4	6	Aplicativo: instável
U	5	15	Falha do sistema: erro de parada
U	5	19	Problema de segurança (Não planejado)
E	5	19	Problema de segurança (Não planejado)
E P	5	19	Problema de segurança (Planejado)
E	5	20	Perda de conectividade de rede (não planejada)
U	6	11	Falha de energia: fio desconectado
U	6	12	Falha de energia: ambiente
P	7	0	Desligamento de API legacy

SORT [/R] [/+n] [/M kilobytes] [/L localidade] [/REC bytes\_de\_registro]  
[[unidade1:][caminho1]nome\_de\_arquivo1] [/T [unidade2:][caminho2]]  
[/O [unidade3:][caminho3] nome\_de\_arquivo3]  
/+n Especifica o n mero do caractere n para  
come ar cada compara o. /+3 indica que  
cada compara o deve come ar no  
terceiro  
antes caractere em cada linha. As linhas com  
menos de n caracteres s o agrupadas  
de outras linhas.  
Por padr o, as compara es come am no  
primeiro caractere em cada linha.  
Substitui a localidade padr o do  
/L[OCALE] localidade sistema  
pela especificada. A localidade ""C"" ,  
substitu da pela sequ ncia de  
agrupamento  
mais r pida e , atualmente a  nica  
alternativa. A classifica o sempre  
diferencia maiusculas de min sculas.  
Especifica a quantidade de mem ria  
/M[EMORY] kilobytes principal  
a ser usada para a classifica o em  
kilobytes. O tamanho da mem ria  
est   sempre restrito ao m nimo de 160 kilobytes. Se  
o tamanho da mem ria for especificado a  
quantidade exata ser usada para a  
classifica o, independentemente da  
quantidade de mem ria principal  
dispon vel.  
ao O melhor desempenho , geralmente obtido  
Por n o especificar um tamanho de mem ria.  
uma padr o, a classifica o ser feita em  
couber passagem (sem arquivo tempor  rio) se  
caso no tamanho de mem ria m  ximo padr o;  
em contrario, a classifica o ser feita  
parcialmente duas passagens (com os dados  
ambas as classificados sendo armazenados em um  
sejam arquivo tempor  rio) de forma que as  
padr o , quantidades de mem ria usadas para  
a passagens de classifica o e mesclagem  
iguais. O tamanho m  ximo de mem ria  
90% da mem ria principal dispon vel se  
entrada e a sa da forem arquivos; caso

principal.  
/REC[ORD\_MAXIMUM] caracteres  
caracteres  
65535).  
/R[EVERSE]  
''  
[unidade1:] [caminho1] arquivo1  
classificado.  
padr o  
como  
/T[TEMPORARY]  
[unidade2:] [caminho2]  
ser  
dados  
padr o ,  
sistema.  
/O[UTPUT]  
[unidade3:] [caminho3] arquivo3 Especifica o arquivo onde  
ser armazenado o  
na  
arquivo de

contrario, ser 45% da memoria  
Especifica o n mero mximo de  
em um registro (padr o 4096; mximo  
Inverte a ordem de classifica o; isto  
classifica de Z a A, e depois de 9 a 0.  
Especifica o arquivo a ser  
Se n o for especificado, a entrada  
ser classificada. A especifica o do  
arquivo de entrada , mais r pida do que  
o redirecionamento do mesmo arquivo  
entrada padr o.  
Especifica o caminho do diret rio onde  
armazenada a classifica o, caso os  
n o caibam na memoria principal. O  
usar um diret rio tempor rio do  
sistema.  
resultado da classifica o. Se n o for  
especificado, os dados ser o gravados  
sa da padr o. A especifica o do  
sa da , mais r pida do que o  
redirecionamento da sa da padr o para o  
mesmo arquivo.

Inicia uma janela separada para executar um programa ou comando especificado.

```
START ["title"] [/D path] [/I] [/MIN] [/MAX] [/SEPARATE | /SHARED]
        [/LOW | /NORMAL | /HIGH | /REALTIME | /ABOVENORMAL | /BELOWNORMAL]
        [/NODE <nó de NUMA>] [/AFFINITY <máscara_afinidade_hexadecimal>]
[/WAIT]
        [/B] [comando/programa] [parâmetros]

    "title"      Título a ser exibido na barra de títulos da janela.
    path         Diretório de início.
    B            Inicia um aplicativo sem criar nova janela. O aplicativo
    tem         o tratamento de ^C ignorado. A menos que o aplicativo
    ative o       processamento de ^C, o comando ^Break é a única forma de
    I            interromper o aplicativo.
    o            O novo ambiente será o ambiente original transferido para
    MIN          cmd.exe e não é o ambiente atual.
    MAX          Inicia a janela minimizada.
    SEPARATE    Inicia um programa do Windows de 16 bits em um espaço de
    memória separado.
    SHARED      Inicia um programa do Windows de 16 bits em um espaço de
    memória compartilhado.
    LOW          Inicia o aplicativo na classe de prioridade IDLE.
    NORMAL       Inicia o aplicativo na classe de prioridade NORMAL.
    HIGH         Inicia o aplicativo na classe de prioridade HIGH.
    REALTIME    Inicia o aplicativo na classe de prioridade REALTIME.
    ABOVENORMAL Inicia o aplicativo na classe de prioridade ABOVENORMAL.
    BELOWNORMAL Inicia o aplicativo na classe de prioridade BELOWNORMAL.
    NODE         Especifica o nó da Arquitetura de Memória Não Uniforme
    (NUMA)       com um inteiro decimal.
    AFFINITY     Especifica a máscara de afinidade do processador com um
    número       hexadecimal. O processo está restrito a execução nesses
                processadores.

    quando       A máscara de afinidade é interpretada de forma diferente
    de          /AFFINITY e /NODE são combinados. Especifique a máscara
    deslocada   afinidade se a máscara do processador do nó NUMA for
    restrito    para a direita para começar no bit zero. O processo é
    de          a execução nesses processadores em comum entre a máscara
    processadores afinidade especificada e o nó NUMA. Se não houver
    NUMA         em comum, o processo estará restrito a execução no nó
                especificado.
    WAIT         Inicia o aplicativo e espera que ele finalize o
                comando/programa.
                Se for um comando cmd interno ou um arquivo em lotes, o
                processador do comando será executado com a opção /K para
```

que o cmd.exe. Isso significa que a janela permanecerá depois comando for executado.

lotes, Se não for um comando cmd interno nem um arquivo em ele será um programa executado como um aplicativo em janela ou um aplicativo em console.

parameters São os parâmetros transferidos para o comando/programa.

OBSERVAÇÃO: As opções SEPARATE e SHARED não são suportadas em plataformas de 64 bits.

A especificação /NODE permite que sejam criados processos de uma forma que aproveite localidade de memória em sistemas NUMA. Por exemplo, dois processos que se comuniquem pesadamente um com o outro por meio de memória compartilhada podem ser criados para compartilhar o mesmo nó NUMA preferencial a fim de minimizar latências de memória. Eles alocam memória do mesmo nó NUMA, quando for possível, e ficam liberados para executar em processadores fora do nó especificado.

```
start /NODE 1 application1.exe  
start /NODE 1 application2.exe
```

Esses dois processos podem ser ainda mais restringidos para executar em processadores específicos dentro do mesmo nó NUMA. No exemplo a seguir, application1 é executado nos dois processadores de ordem inferior do nó, enquanto application2 é executado nos próximos dois processadores do nó. Esse exemplo presume que o nó especificado tem no mínimo quatro processadores lógicos. Observe que o número do nó pode ser alterado para qualquer número de nó válido para aquele computador sem precisar alterar a máscara de afinidade.

```
start /NODE 1 /AFFINITY 0x3 application1.exe  
start /NODE 1 /AFFINITY 0xc application2.exe
```

Se as extensões de comando estiverem ativadas, a invocação de comando externa através da linha de comando ou do comando START será alterada como a seguir:

Arquivos não executáveis podem ser chamados através de sua associação de arquivos apenas ao digitar o nome do arquivo como um comando. (e.g. WORD.DOC iniciaria o aplicativo associado com a extensão de arquivo .DOC). Consulte os comandos ASSOC e FTYPE para obter informações sobre como criar essas associações a partir de um script de comando.

Ao executar um aplicativo que seja um aplicativo GUI de 32 bits, o CMD.EXE

não espera o aplicativo terminar para voltar ao prompt de comando. Esse novo comportamento NÃO ocorre se estiver sendo executado a partir de um script de comando.

Ao executar uma linha de comando cujo primeiro token seja a cadeia de caracteres "CMD " sem uma extensão ou qualificador de caminhos, o "CMD"  
é substituído pelo valor da variável COMSPEC. Isso evita selecionar o  
\*  
CMD.EXE na pasta atual.

Ao executar uma linha de comando cujo primeiro token NÃO contenha uma extensão, o CMD.EXE usará o valor da variável de ambiente PATHEXT para determinar quais extensões serão procuradas e em que ordem. O valor padrão da variável PATHEXT é:

.COM; .EXE; .BAT; .CMD

Observe que a sintaxe é a mesma da variável PATH, com ponto-e-vírgula separando os diferentes elementos.

Ao procurar por um executável, se não houver correspondência em nenhuma extensão, ele verifica se o nome corresponde a um nome de pasta. Se corresponder, o comando START inicia o Explorer nesse caminho. Se for feito a partir da linha de comando, será o equivalente a fazer CD /D para esse caminho.

Associa um caminho a uma letra de unidade.

```
SUBST [unidade1: [unidade2:]caminho]
SUBST unidade1: /D
```

```
unidade1:      Especifica a unidade virtual a que se atribui o caminho.
[unidade2:]caminho Unidade física e caminho que se deseja atribuir
                   a uma unidade virtual.
/D                 Exclui uma unidade substituída (virtual).
```

Digite SUBST sem parâmetros para exibir a lista das unidades virtuais atuais.

```
SYSTEMINFO [/S sistema [/U usuário [/P [senha]]]] [/FO formato] [/NH]
```

**Descrição:**

Esta ferramenta exibe informações de configuração de sistema para um computador local ou remoto, inclusive níveis de service pack.

**Lista de parâmetros:**

/S	system	Especifica o sistema remoto ao qual se conectar.
/U	[domínio\]usuário	Especifica o contexto de usuário em que o comando deve ser executado.
/P	[senha]	Especifica a senha para o contexto de usuário. Solicita entrada se omitido.
/FO	format	Especifica o formato em que a saída deve ser exibida. Valores válidos: "TABLE", "LIST", "CSV".
/NH não		Especifica que o "cabeçalho de coluna" deve ser exibido na saída. Válido apenas para formatos "TABLE" e "CSV".
/?		Exibe esta mensagem da Ajuda.

**Exemplos:**

```
SYSTEMINFO  
SYSTEMINFO /?  
SYSTEMINFO /S sistema  
SYSTEMINFO /S sistema /U usuário  
SYSTEMINFO /S sistema /U domínio\usuário /P senha /FO TABLE  
SYSTEMINFO /S sistema /FO LIST  
SYSTEMINFO /S sistema /FO CSV /NH
```

```
TASKKILL [/S sistema [/U usuário [/P [senha]]]]
{ [/FI filtro] [/PID processid | /IM imagename] } [/T] [/F]
```

#### Descrição:

Esta ferramenta é usada para finalizar tarefas por identificação de processo (PID) ou nome de imagem.

#### Lista de parâmetros:

/S system	Especifica o sistema remoto ao qual se conectar.
/U [domínio\]usuário	Especifica o contexto de usuário em que o comando deve ser executado.
/P [senha]	Especifica a senha para o contexto de usuário. Solicita entrada, caso omitido.
/FI filtro	aplica um filtro para selecionar um conjunto de tarefas. Permite que "*" seja usado. Por exemplo: nome_imagem eq acme*
/PID processid	especifica a PID do processo a ser finalizado. Use TaskList para obter a PID.
/IM imagename	Especifica o nome de imagem do processo a ser finalizado. O caractere curinga '*' pode ser usado para especificar todas as tarefas ou nomes de imagem.
/T	Finaliza o processo especificado e quaisquer processos filho iniciados por ele.
/F	Especifica a finalização forçada do(s) processo(s).
/?	Exibe esta mensagem da Ajuda.

#### Filtro(s):

Nome do filtro	Operadores válidos	Valor(es) válido(s)
<hr/>		
- STATUS	eq, ne	EM EXECUÇÃO   NÃO RESPONDENDO
<hr/>		
DESCONHECIDO		
IMAGENAME	eq, ne	Qualquer nome de imagem.
PID	eq, ne, gt, lt, ge, le	Valor de PID.
SESSION	eq, ne, gt, lt, ge, le	Número de sessão.
CPUTIME	eq, ne, gt, lt, ge, le	Horário de CPU no formato de hh:mm:ss. hh - horas, mm - minutos, ss - segundos
<hr/>		
MEMUSAGE	eq, ne, gt, lt, ge, le	Uso de memória em KB.
USERNAME	eq, ne	Nome de usuário em formato [domínio\]usuário
MODULES	eq, ne	Nome da DLL
SERVICES	eq, ne	Nome de serviço
WINDOWTITLE	eq, ne	Título de janela

OBS.

----

- 1) O caractere curinga '\*' para a chave /IM é aceito somente quando um filtro é aplicado.
- 2) O encerramento de processos remotos sempre é forçado (/F).
- 3) os filtros "WINDOWTITLE" e "STATUS" não são considerados quando um computador remoto é especificado.

Exemplos:

```
TASKKILL /IM notepad.exe
TASKKILL /PID 1230 /PID 1241 /PID 1253 /T
TASKKILL /F /IM cmd.exe /T
TASKKILL /F /FI "PID ge 1000" /FI "WINDOWTITLE ne untitled*"
TASKKILL /F /FI "USERNAME eq NT AUTHORITY\SYSTEM" /IM notepad.exe
TASKKILL /S sistema /U domínio\usuário /FI "USERNAME ne NT*" /IM *
TASKKILL /S sistema /U usuário /P senha /FI "IMAGENAME eq note*"
```

```
TASKLIST [/S sistema [/U nome_usuário [/P [senha]]]
          [/M [módulo] | /SVC | /V] [/FI filtro] [/FO formato] [/NH]
```

**Descrição:**

Esta ferramenta exibe uma lista de aplicativos em execução no momento em um computador local ou remoto.

**Lista de parâmetros:**

/S	sistema	Especifica o sistema remoto ao qual se conectar.
/U	[domínio\]usuário	Especifica o contexto de usuário em que o comando deve ser executado.
/P	[senha]	Especifica a senha para o usuário. Solicita entrada se omitido.
/M	[módulo] nome	Lista todas as tarefas que estão usando o exe/dll. Se o nome do módulo não for especificado, todos os módulos carregados são exibidos.
/SVC	processo.	Exibe os serviços hospedados em cada
/APPS		Exibe os Aplicativos da Loja e seus processos associados.
/V		Exibe informações detalhadas da tarefa.
/FI	filtro	Exibe um conjunto de tarefas correspondentes a critérios especificados pelo filtro.
/FO	formato	Especifica o formato de saída. Valores válidos: "TABLE", "LIST", "CSV".
/NH	deve	Especifica que o "Cabeçalho de Coluna" não ser exibido na saída. Válido apenas para os formatos "TABLE" e "CSV".
/?		Displays this help message.

**Filtros:**

Nome do Filtro	Operadores Válidos	Valor(es) Válido(s)
STATUS	eq, ne	EMEXECUÇÃO   SUSPENSO NÃO RESPONDE   DESCONHECIDO
IMAGENAME	eq, ne	Nome da imagem
PID	eq, ne, gt, lt, ge, le	Valor do PID
SESSION	eq, ne, gt, lt, ge, le	Número da sessão
SESSIONNAME	eq, ne	Nome da sessão
CPUTIME	eq, ne, gt, lt, ge, le	Tempo de CPU no formato de hh:mm:ss. hh - horas,

MEMUSAGE	eq, ne, gt, lt, ge, le	mm - minutos, ss - segundos
USERNAME	eq, ne	Uso da memória em KB
SERVICES	eq, ne	Nome de usuário no formato [domínio\]usuário
WINDOWTITLE	eq, ne	Nome do serviço
MODULES	eq, ne	Título da janela
		DLL name

OBSERVAÇÃO: não há suporte aos filtros "WINDOWTITLE" e "STATUS" quando um computador remoto é consultado.

#### Examples:

```

TASKLIST
TASKLIST /M
TASKLIST /V /FO CSV
TASKLIST /SVC /FO LIST
TASKLIST /APPS /FI "STATUS eq EM EXECUÇÃO"
TASKLIST /M wbem*
TASKLIST /S sistema /FO LISTA
TASKLIST /S sistema /U domínio\nomedesusário /FO CSV /NH
TASKLIST /S sistema /U nomedeusuário /P senha /FO TABELA /NH
TASKLIST /FI "NOMEDEUSUÁRIO ne AUTORIDADE NT\SISTEMA" /FI "STATUS eq
em execução"
```

Exibe ou define a hora do sistema.

TIME [/T | hora]

Digite TIME sem parâmetros para exibir a configuração de hora atual e poder  
digitar a nova hora. Pressione ENTER para manter a hora inalterada.

Se as extensões de comando estiverem ativadas, o comando TIME dará suporte à  
opção /T que informa ao comando para exibir apenas a hora atual, sem solicitar  
uma nova hora.

Define o título da janela do prompt de comando.

TITLE [cadeia de caracteres]

string      Especifica o título da janela do prompt de comando.

Exibe de forma gr fica a estrutura de pastas de uma unidade ou caminho.

TREE [unidade:] [caminho] [/F] [/A]

/F Exibir os nomes dos arquivos de cada pasta.

/A Usar ASCII em vez de caracteres estendidos.

Exibe o conteúdo de um ou mais arquivos de texto.

TYPE [unidade:] [caminho]nomedearquivo

Exibe a versão do Windows.

VER

Faz com que o cmd.exe verifique ou não se seus arquivos foram gravados corretamente no disco.

VERIFY [ON | OFF]

Digite VERIFY sem parâmetros para exibir a configuração atual de VERIFY.

Exibe o nome e o número de série do disco, caso existam.

VOL [unidade:]

[opções globais] <comando>

As seguintes opções globais estão disponíveis:	
/NAMESPACE	Caminho do namespace em relação ao qual o alias opera.
/ROLE	Caminho da função que contém as definições de alias.
/NODE	Servidores em relação aos quais o alias operar.
/IMPLEVEL	Nível de representação de cliente.
/AUTHLEVEL	Nível de autenticação do cliente.
/LOCALE	Identificação do idioma a ser usada pelo cliente.
/PRIVILEGES	Ativar ou desativar todos os privilégios.
/TRACE	Envia informações de depuração para stderr.
/RECORD	Registra todos os comandos de entrada e saída.
/INTERACTIVE	Define ou redefine o modo interativo.
/FAILFAST	Define ou redefine o modo FailFast.
/USER	Usuário a ser usado durante a sessão.
/PASSWORD	Senha para fazer logon na sessão.
/OUTPUT	Especifica o modo para redirecionamento de saída.
APPEND	Especifica o modo para redirecionamento de saída.
/AGGREGATE	Define ou redefine o modo agregado.
/AUTHORITY	Especifica o <tipo de autoridade> para a conexão.
/?[:<BRIEF FULL>]	Informações sobre o uso.

Para obter mais informações sobre um tipo de opção global específico, digite: opção-nome /?

Os seguintes aliases estão disponíveis na função atual:

ALIAS	- Acesso aos aliases disponíveis no sistema
local	
BASEBOARD	- Gerenciamento de placa base (também conhecida como placa-mãe ou placa do sistema).
como placa-mãe ou placa do sistema).	
BIOS	- Gerenciamento de BIOS (Basic input/output services).
services).	
BOOTCONFIG	- Gerenciamento de configuração de inicialização.
inicialização.	
CDROM	- Gerenciamento de CD-ROM.
COMPUTERSYSTEM	- Gerenciamento do sistema do computador.
CPU	- Gerenciamento de CPU.
CSPRODUCT	- Informações sobre produtos do sistema de computador de SMBIOS.
computador de SMBIOS.	
DATAFILE	- Gerenciamento de DataFile.
DCOMAPP	- Gerenciamento de Aplicativo DCOM.
DESKTOP	- Gerenciamento da área de Trabalho do Usuário.
DESKTOPMONITOR	- Gerenciamento de Monitor da área de Trabalho.
DEVICEMEMORYADDRESS	- Gerenciamento de endereços de memória de dispositivo.
dispositivo.	
DISKDRIVE	- Gerenciamento de unidades de disco físicas.
DISKQUOTA	- Uso de espaço em disco para volumes NTFS.
DMACHANNEL	- Gerenciamento de canal de DMA (Direct memory access).
access).	
ENVIRONMENT	- Gerenciamento de configurações de ambiente do sistema.
sistema.	
FSDIR	- Gerenciamento de entrada de diretório de Filesystem.
Filesystem.	
GROUP	- Gerenciamento de contas de grupo.
IDECONTROLLER	- Gerenciamento de Controlador IDE.
IRQ	- Gerenciamento de IRQ.

JOB - Fornece acesso aos trabalhos agendados usando o serviço de agendamento.  
 LOADORDER - Gerenciamento de serviços do sistema que definem dependências de execução.  
 LOGICALDISK - Gerenciamento de dispositivo de armazenamento local.  
 LOGON - Sessões de LOGON.  
 MEMCACHE - Gerenciamento de memória cache.  
 MEMORYCHIP - Informações de chip de memória.  
 MEMPHYSICAL - Gerenciamento de memória física do sistema de computador.  
 NETCLIENT - Gerenciamento de Cliente de Rede.  
 NETLOGIN - Gerenciamento de informações de logon na rede (em relação a um usuário específico).  
 NETPROTOCOL - Gerenciamento de protocolos (e suas características de rede).  
 NETUSE - Gerenciamento de conexão de rede ativa.  
 NIC - Gerenciamento do Controlador de Interface de Rede (NIC).  
 NICCONFIG - Gerenciamento de adaptadores de rede.  
 NTDOMAIN - Gerenciamento de Domínio NT.  
 NTEVENT - Entradas no Log de Eventos do NT.  
 NTEVENTLOG - Gerenciamento de arquivo de log de eventos NT.  
 ONBOARDDEVICE - Gerenciamento de dispositivos de adaptador comuns internos da placa-mãe (placa do sistema).  
 OS - Gerenciamento de Sistemas Operacionais instalados.  
 PAGEFILE - Gerenciamento de troca de arquivo da memória virtual.  
 PAGEFILESET - Gerenciamento de configurações de arquivo de paginação.  
 PARTITION - Gerenciamento de reas particionadas de um disco físico.  
 PORT - Gerenciamento de porta de E/S.  
 PORTCONNECTOR - Gerenciamento de portas de conexão física.  
 PRINTER - Gerenciamento de dispositivos de impressora.  
 PRINTERCONFIG - Gerenciamento de configuração de dispositivo de impressora.  
 PRINTJOB - Gerenciamento de trabalhos de impressão.  
 PROCESS - Gerenciamento de processos.  
 PRODUCT - Gerenciamento de tarefas de pacote de instalação.  
 QFE - Quick Fix Engineering.  
 QUOTASETTING - Definindo informações de cotas de disco em um volume.  
 RDACCOUNT - Gerenciamento de permissões de conexão da área de Trabalho Remota.  
 RDNIC - Gerenciamento de conexão da área de Trabalho Remota em um adaptador de rede específico.  
 RDPERMISSIONS - Permissões para uma conexão específica da área de Trabalho Remota.  
 RDToggle - Ativando ou desativando remotamente o ouvinte da área de Trabalho Remota.  
 RECOVEROS - Informações que serão coletadas da memória quando o sistema operacional falhar.  
 REGISTRY - Gerenciamento do Registro do sistema do computador.  
 SCSICONTROLLER - Gerenciamento de Controlador SCSI.  
 SERVER - Gerenciamento de informações de servidor.  
 SERVICE - Gerenciamento de aplicativos de serviço.

SHADOWCOPY	- Gerenciamento de c?pia de sombra.
SHADOWSTORAGE	- Gerenciamento de rea de armazenamento de c?pia de sombra.
SHARE	- Gerenciamento de recursos compartilhados.
SOFTWAREELEMENT	- Gerenciamento dos elementos de um produto de software instalado em um sistema.
SOFTWAREFEATURE	- Gerenciamento de subconjuntos de produtos de software de SoftwareElement.
SOUNDDEV	- Gerenciamento de Dispositivos de Som.
STARTUP	- Gerenciamento de comandos que s?o executados automaticamente quando os usu?rios fazem logon no sistema de computador.
SYSACCOUNT	- Gerenciamento de conta do sistema.
SYSDRIVER	- Gerenciamento do driver do sistema para um servi?o de base.
SYSTEMENCLOSURE	- Gerenciamento de compartimento f?sico do sistema.
SYSTEMSLOT	- Gerenciamento de pontos de conex?o f?sicos, inclusive portas, slots e perif?ricos e de pontos de conex?o patenteados.
TAPEDRIVE	- Gerenciamento de unidade de fita.
TEMPERATURE	- Gerenciamento de dados de um sensor de temperatura (term?metro eletr?nico).
TIMEZONE	- Gerenciamento de dados de fuso hor?rio.
UPS	- Gerenciamento de no-break.
USERACCOUNT	- Gerenciamento de contas de usu?rio.
VOLTAGE	- Gerenciamento de dados de sensor de voltagem (volt?metro eletr?nico).
VOLUME	- Gerenciamento de armazenamento de volume local.
VOLUMEQUOTASETTING	- Associa a configura?o de cota de disco a um volume de disco espec?fico.
VOLUMEUSERQUOTA	- Gerenciamento de cota de volume de armazenamento por usu?rio.
WMISET	- Gerenciamento de par?metros operacionais do servi?o WMI.

Para obter mais informa?es em um alias espec?fico, digite: alias /?

CLASS	- Sai para o esquema WMI completo.
PATH	- Sai para caminhos completos de objetos WMI.
CONTEXT	- Exibe o estado de todas as op?es globais.
QUIT/EXIT	- Sai do programa.

Para obter mais informa?es sobre CLASS/PATH/CONTEXT, digite: (CLASS | PATH | CONTEXT) /?

Copia arquivos e rvores de diret\$rios.

```
XCOPY origem [destino] [/A | /M] [/D[:data]] [/P] [/S [/E]] [/V] [/W]
                           [/C] [/I] [/Q] [/F] [/L] [/G] [/H] [/R] [/T]
                           [/U]
                           [/K] [/N] [/O] [/X] [/Y] [/‐Y] [/Z] [/B] [/J]
                           [/EXCLUDE:arquivo1[+arquivo2][+arquivo3]...]
```

origem Especifica o(s) arquivo(s) a ser(em) copiado(s).  
destino Especifica o local e/ou o nome dos novos arquivos.  
/A Copia somente arquivos com o atributo de arquivamento definido; nao altera o atributo.  
/M Copia somente arquivos com o atributo de arquivamento definido; desativa o atributo de arquivamento.  
/D:m-d-a Copia os arquivos alterados durante ou aps a data especificada. Se nao for definida uma data, copia apenas os arquivos cujo tempo de origem seja mais novo do que o tempo de destino.  
/EXCLUDE:arquivo1[+arquivo2][+arquivo3]... Especifica uma lista de arquivos contendo cadeias de caracteres. Cada cadeia deve estar em uma linha separada nos arquivos. Quando qualquer cadeia corresponder a qualquer parte do caminho absoluto do arquivo a ser copiado, esse arquivo deixar de ser copiado. Por exemplo, especificar uma cadeia de caracteres como \obj\ ou .obj excluir todos os arquivos sob o diret\$rio obj ou com a extensao .obj, respectivamente.  
/P Solicita confirma\$ao antes de criar cada arquivo de destino.  
/S Copia diret\$rios e subdiret\$rios, exceto os vazios.  
/E Copia diret\$rios e subdiret\$rios, inclusive os vazios.  
O mesmo que /S /E. Pode ser usado para modificar /T.  
/V Verifica o tamanho de cada novo arquivo.  
/W Solicita que voc^ pressione uma tecla antes de copiar.  
/C Continua copiando, mesmo que ocorram erros.  
/I Se o destino nao existir e mais de um arquivo estiver sendo copiado, pressupoe que o destino deve ser um diret\$rio.  
/Q Nao exibe os nomes de arquivos ao copiar.  
/F Exibe os nomes de arquivos de origem e de destino completos ao copiar.  
/L Exibe arquivos que seriam copiados.  
/G Permite a c\$opia de arquivos criptografados para um destino que nao oferece suporte a criptografia.  
/H Copia arquivos ocultos e do sistema tamb,m.  
/R Substitui arquivos somente leitura.  
/T Cria a estrutura de diret\$rios, mas nao copia arquivos.  
Nao inclui diret\$rios ou subdiret\$rios vazios. /T /E inclui diret\$rios e subdiret\$rios vazios.  
/U Copia apenas os arquivos que j existem no destino.  
/K Copia atributos. O Xcopy normal redefinir os atributos somente leitura.

/N	Copia usando os nomes curtos gerados.
/O	Copia as informações de ACL e proprietário do arquivo.
/X	Copia configurações de auditoria de arquivo (implica /O).
/Y	Suprime o prompt para você confirmar se deseja substituir um arquivo de destino existente.
/-Y	Exibe o prompt para você confirmar se deseja substituir um arquivo de destino existente.
/Z	Copia arquivos de rede no modo reiniciável.
/B	Copia o próprio Vínculo Simbólico, e não o destino do vínculo.
/J	Copia usando E/S sem buffer. Recomendável para arquivos muito grandes.
muito	

A opção /Y pode ser predefinida na variável de ambiente COPYCMD. Isso pode ser anulado por /-Y na linha de comando.