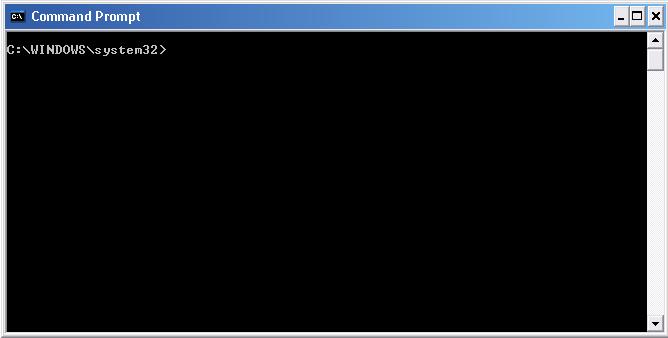
Batch-Dictionary

# Overview:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| %var% | Color | FC | Msg | Set | User/User32 |
| %var:~x,y% | Command.com | Find | Msgbox | Setlocal | *Variables* |
| %var:a=b% | Comp | Findstr | Net | Shift | Ver |
| %0 | Compact | For | Netsh | *Shortnames* | Wmic |
| >nul | Convert | Format | Netstat | Shutdown | Xcopy |
| >”Path” | Copy | Ftp | Path | Sndrec32 |  |
| 2>&1 | Date | Ftype | Pause | Sort |  |
| *Arguments* | Defrag | Goto | Ping | *Spring-points* |  |
| *Ascii* | Del / Erase | Gpresult | Ping localhost | Start |  |
| Assoc | Dir | Help | *Program-paths* | *Symbols* |  |
| Attrib | Driverquery | Iexplore | Prompt | Systeminfo |  |
| Break | Echo | If | Pushd / Popd | Taskkill |  |
| Cacls | Edit | Label | Rd / Rmdir | Tasklist |  |
| Call | Endlocal | Md / Mkdir | Reg | Time |  |
| Cd / Chdir | Erase / Del | Mode | Rem | Title |  |
| Chkdsk | *Errorlevel* | More | Ren | Tree |  |
| Cls | Exit | Move | Runas | Tskill |  |
| Cmd | Explorer | Mplay32 | Sc | Type |  |

* **Bold**, *italic*, underlined Words exist as an extra-point (e.g. ***Variables***).
* On the last page there is an ordering for beginners.



# Explanations

* **%var%:**

A string enclosed by two **%-characters** is a **variable** that was set to something specific before. This is done by the ***SET*-command** (more about this read SET).

By using the command "SETLOCAL ENABLEDELAYEDEXPANSION" variables can also be set and called **within parentheses**. In this case they have to be enclosed by two **!-characters**. More about this read "***Variables***".

* **%var:~x,y% :**

This is a very complicated type of a variable. Usually there would be written %var%. The **addition :~x,y** can indicate, for which characters of the variable a specific command (e.g. echo) shall be executed. **x** is the sign that should **be started from**. **y** is the sign, where should be stopped at. Is y **negative**, it will be counted from the back.

Examples:

echo %time:~0,2% **The** current time will be displayed, but only the first 2 numbers

started from the beginning (0). These are the current hours.

echo %time:~3,2% **The** current time will be displayed, but only the first 2 numbers started from the third sign. These are the current minutes.

The time is usually given as HH/MM/SS,MS. The slashes and the

comma are characters, too. The command starts from the third

sign (HH/). After two signs it stops (MM).

echo %time%:~0,-2% **Every** digit until the second to last will be displayed.

The whole thing goes with letters instead of numbers, too:

echo %cd:~0,20% **The** first 20 letters and characters will be printed.

* **%var:a=b% :**

This is another form of a variable. The normal variable is %var%. The **addition :a=b**, though, causes that **string „a“**, which consists of any amount of letters, numbers or symbols (except % or =), is replaced by the **string „b”.** All strings „a“ are replaced.

Examples:

Set Stone=This will be a variable.

Echo %Stone%

Set Stone=%Stone:will be=is%

Echo %Stone%

Output:

This will be a variable.

This is a variable.

🡪 The words “will be” are replaced by “is”. “will be” is string a in this case, “is” is b:

%Stone:a=b% --- %Stone:will be=is%

* **%0:**

%0 is in general an ***argument***.

%0 is the currently opened **file** under its **path**. E.G. %0 here would give the path of this file.

This can be used to change or restart the current file.

Example:

call %0

This command starts the current file again.

%0 changes if you call a ***spring point*** by the ***call***-command. In this case %0 changes to the name of that point.

* **>nul:**

The addition "**>nul**" allows you to **suppress** the **display of the progress** of a command, that means the output is hidden. Errors are still shown, though. To pass them by, too, you have to add "***2>&1***".

"**2>nul**" as suppresses the output of errors as well, but not the main output.

"**>nul**" and "**2>nul**" can be used together, but they do not have to.

Examples:

Input: pause

Output: Press any key to continue. . .

Input: pause >nul

Output:

Input: ***find***

Output: FIND: Parameter format wrong

Input: find >nul

Output: FIND: Parameter format wrong

Input: find 2>nul

Output:

Input: ***taskkill*** /IM taskmgr.exe >nul 2>nul

Output:

* In this case the output will be suppressed in case of success AND failure.
* **>"Path"**

This addition can be added to most of the commands. The output of the command will be saved into the file behind the >.

Thus "***tasklist*** /SVC**>**"%userprofile%\Desktop\Processlist.txt"" would save all the currently running processes into Processlist.txt on the desktop.

This result can also be reached by using the ***FOR***-command, but it takes more time and is more complicated, though.

FOR /F "delims=" %%A IN ('tasklist /SVC') DO echo %%A**>>**"%userprofile%\Desktop\Processlist.txt"

Because of the command "wmic … get" cannot be used inside the FOR-command, you have to save it by using the addition >"Path" into a file. This file can be analyzed by the FOR-command, then:

Wmic process get CommandLine,ProcessId**>**Processes.log

FOR /F "tokens=1-8 delims=\. " %%A IN (Processes.log) DO if /I "%%D" == "svchost" echo %%A\%%B\%%C\%%D.%%E %%F %%G %%H

REM C:\Windows\system32\svchost.exe –k imgsvc *PID*

REM %A %B %C %D %E %F %G %H

* **2>&1:**

The addition "2>&1" can only be used behind "***>nul***" and **hides errors from the output** of a command. This effect can also be done by "**2>nul**", but this works also without "***>nul***". More about this read "***>nul***".

Examples:

Input : ***del*** "C:\Windows\NotExistingFile.txt"

Output : C:\Windows\NotExistingFile.txt was not found.

Input : del "C:\Windows\NotExistingFile.txt" 2>&1

Output : C:\Windows\NotExistingFile.txt was not found.

Input : del "C:\Windows\NotExistingFile.txt" >nul

Output : C:\Windows\NotExistingFile.txt was not found.

Input : del "C:\Windows\NotExistingFile.txt" >nul 2>&1

Output :

# A

* **Arguments:**

Arguments are strings, which are used while **starting a file** as extra commands.

Arguments are ***%0* to %9**. Examples of starting a file with arguments:

***start*** batch-Dictionary-Helper.bat "This will be printed, if there is written ' echo %1 ' in the file."

Multiple words with spaces can be summarized to one argument if they are written in

" " as given in the example. Here’s the example a bit more detailed:

Content of the bat-file "batch-Dictionary-Helper.bat":

@echo off

echo %1 , %2 , %3 , %4

pause >nul

Command in CMD:

**call** "batch-Dictionary-Helper.bat" "Argument No. 1" "Argument No. 2" etc. "will be printed."

This would be displayed:

Argument No. 1 , Argument No. 2 , etc. , will be printed.

( %1 %2 %3 %4 )

Arguments can also be used by calling ***spring points***:

@echo off

set Variable=do a lot.

call **:Springpunkt3** With this you can "%Variable%"

pause >nul

:Springpunkt3

echo %1 %2 %3 %4 %5

goto :PointDoesNotExist

This would be displayed:

With this you can do a lot.

(%1 %2 %3 %4 %5 )

You can also expand arguments. The expansions are explained in the section „***FOR***“ (e.g. %~nx1).

Examples:

***Set*** PathOfTheFile=%~s1

🡪 This would set the variable to a ***shortname***.

***If*** /i "%~n1" == "Virus" echo The file name is "Virus"!

* **Ascii:**

#################################################################

# #

# The Ascii-codes below are entered as numbers into notepad #

# and have been running in CMD. The batch-file has to be saved #

# as ANSI, because Unicode is unreadable. #

# #

#################################################################

Ascii-codes are written as the following:

**Alt + *number on the numeric keypad***

You hold Alt and enter the following numbers on the **numeric keypad**:

|  |  |  |  |
| --- | --- | --- | --- |
| 140 | ¯ | 212 | ╚ |
| 143 | ┼ | 214 | ═ |
| 144 | ╔ | 215 | ╬ |
| 148 | ÷ | 222 | ╠ |
| 150 | ¹ | 223 | » |
| 151 | ¨ | 225 | ▀ |
| 154 | ▄ | 229 | big ¹ |
| 163 | · | 231 | ■ |
| 166 | ¬ | 233 | ┌ |
| 167 | ║ | 234 | █ |
| 168 | ┐ | 235 | ┘ |
| 169 | « | 236 | ² |
| 172 | ╝ | 239 | ┤ |
| 175 | ╗ | 240 | ¡ |
| 181 | ┴ | 241 | ▒ |
| 182 | ┬ | 247 | © |
| 183 | └ | 248 | ░ |
| 184 | (R) | 249 | ¿ |
| 199 | ├ | 251 | ╣ |
| 210 | ╩ | 252 | │ |
| 211 | ╦ | 253 | ▓ |

A bit more organized:

|  |  |  |  |
| --- | --- | --- | --- |
| 144 | ╔ | 233 | ┌ |
| 212 | ╚ | 183 | └ |
| 175 | ╗ | 168 | ┐ |
| 172 | ╝ | 235 | ┘ |
| 214 | ═ | Minus sign | - |
| 215 | ╬ | 143 | ┼ |
| 222 | ╠ | 199 | ├ |
| 167 | ║ | 252 | │ |
| 251 | ╣ | 239 | ┤ |
| 210 | ╩ | 181 | ┴ |
| 211 | ╦ | 182 | ┬ |
| 148 | ÷ | 225 | ▀ |
| 163 | · | 231 | ■ |
| 151 | ¨ | 154 | ▄ |
| 150 | ¹ | 234 | █ |
| 229 | big ¹ | 253 | ▓ |
| 236 | ² | 241 | ▒ |
| 223 | » | 248 | ░ |
| 169 | « | 166 | ¬ |
| 247 | © | 240 | ¡ |
| 184 | (R) | 249 | ¿ |

Examples:

1. Double lines.

144 - 214 - 214 - ... - 214 - 214 - 175

167 167

167 Text Text Text Text Text Text 167

167 167

212 - 214 - 214 - ... - 214 - 214 - 172

2. In the middle filled spaces.

154 - 231 - 231 - ... - 231 - 231 - 154

234 234

234 Text Text Text Text Text Text 234

234 234

225 - 231 - 231 - ... - 231 - 231 - 225

3. At the edge full spaces, above the upper spaces, below the lower spaces.

234 - 225 - 225 - ... - 225 - 225 - 234

234 234

234 Text Text Text Text Text Text 234

234 234

234 - 154 - 154 - ... - 154 - 154 - 234

4. Single lines.

233 - Many minus signs - 168

252 252

252 Text Text Text Text Text Text 252

252 252

183 - Many minus signs - 235

You can also **add onto every code a zero**. In this case the characters are more organized. The following sings are printed in CMD. The Ascii-codes are entered into notepad. For example, 0134 is † (Jesus cross) in Notepad, but in CMD is printed an å (a with a circle above).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 01:☺ | 045: - | 089: Y | 0133: à | 0177: ▒ | 0221: ¦ |
| 02:☻ | 046: . | 090: Z | 0134: å | 0178: ▓ | 0222: Ì |
| 03: ♥ | 047: / | 091: [ | 0135: ç | 0179: │ | 0223: ▀ |
| 04: ♦ | 048: 0 | 092: \ | 0136: ê | 0180: ┤ | 0224: Ó |
| 05: ♣ | 049: 1 | 093: ] | 0137: ë | 0181: Á | 0225: ß |
| 06: ♠ | 050: 2 | 094: ^ | 0138: è | 0182: Â | 0226: Ô |
| 07: {BEL} (Alert) | 051: 3 | 095: \_ | 0139: ï | 0183: À | 0227: Ò |
| 08: {BS} (Backspace) | 052: 4 | 096: ` | 0140: î | 0184: © | 0228: õ |
| 09: (Tab) | 053: 5 | 097: a | 0141: ì | 0185: ╣ | 0229: Õ |
| 010: {LF} (Enter) | 054: 6 | 098: b | 0142: Ä | 0186: ║ | 0230: µ |
| 011: ♂ | 055: 7 | 099: c | 0143: Å | 0187: ╗ | 0231: þ |
| 012: ♀ | 056: 8 | 0100: d | 0144: É | 0188: ╝ | 0232: Þ |
| 013: {CR} (Enter without effect) | 057: 9 | 0101: e | 0145: æ | 0189: ¢ | 0233: Ú |
| 014: ♫ | 058: : | 0102: f | 0146: Æ | 0190: ¥ | 0234: Û |
| 015: ☼ | 059: ; | 0103: g | 0147: ô | 0191: ┐ | 0235: Ù |
| 016: ► | 060: < | 0104: h | 0148: ö | 0192: └ | 0236: ý |
| 017: ◄ | 061: = | 0105: i | 0149: ò | 0193: ┴ | 0237: Ý |
| 018: ↕ | 062: > | 0106: j | 0150: û | 0194: ┬ | 0238: ¯ |
| 019: ‼ | 063: ? | 0107: k | 0151: ù | 0195: ├ | 0239: ´ |
| 020: ¶ | 064: @ | 0108: l | 0152: ÿ | 0196: ─ | 0240: ­ |
| 021: § | 065: A | 0109: m | 0153: Ö | 0197: ┼ | 0241: ± |
| 022: ▬ | 066: B | 0110: n | 0154: Ü | 0198: ã | 0242: ‗ |
| 023: ↨ | 067: C | 0111: o | 0155: ø | 0199: Ã | 0243: ¾ |
| 024: ↑ | 068: D | 0112: p | 0156: £ | 0200: ╚ | 0244: ¶ |
| 025: ↓ | 069: E | 0113: q | 0157: Ø | 0201: ╔ | 0245: § |
| 026: (Sub-command, like "(" ) | 070: F | 0114: r | 0158: × | 0202: ╩ | 0246: ÷ |
| 027: ← | 071: G | 0115: s | 0159: ƒ | 0203: ╦ | 0247: ¸ |
| 028: ∟ | 072: H | 0116: t | 0160: á | 0204: ╠ | 0248: ° |
| 029: ↔ | 073: I | 0117: u | 0161: í | 0205: ═ | 0249: ¨ |
| 030: ▲ | 074: J | 0118: v | 0162: ó | 0206: ╬ | 0250: · |
| 031: ▼ | 075: K | 0119: w | 0163: ú | 0207: ¤ | 0251: ¹ |
| 032: (Spacebar) | 076: L | 0120: x | 0164: ñ | 0208: ð | 0252: ³ |
| 033: ! | 077: M | 0121: y | 0165: Ñ | 0209: Ð | 0253: ² |
| 034: " | 078: N | 0122: z | 0166: ª | 0210: Ê | 0254: ■ |
| 035: # | 079: O | 0123: { | 0167: º | 0211: Ë | 0255: |
| 036: $ | 080: P | 0124: | | 0168: ¿ | 0212: È |  |
| 037: % | 081: Q | 0125: } | 0169: ® | 0213: ı |  |
| 038: & | 082: R | 0126: ~ | 0170: ¬ | 0214: Í |  |
| 039: ' | 083: S | 0127: ⌂ | 0171: ½ | 0215: Î |  |
| 040: ( | 084: T | 0128: Ç | 0172: ¼ | 0216: Ï |  |
| 041: ) | 085: U | 0129: ü | 0173: ¡ | 0217: ┘ |  |
| 042: \* | 086: V | 0130: é | 0174: « | 0218: ┌ |  |
| 043: + | 087: W | 0131: â | 0175: » | 0219: █ |  |
| 044: , | 088: X | 0132: ä | 0176: ░ | 0220: ▄ |  |

* **assoc:**

This command allows you to **change the file type of a** particular **file extension**. This can make extensions unusable!

Example:

assoc .bat=ERROR

This makes every .bat-file unusable, because the normal file type "batfile" is **overwritten**. Most of the files have the same **association** as their extension adding "file":

.bat=batfile

.txt=txtfile

! .jpg=jpegfile !

! .zip=CompressedFolder !

! .bmp=Paint.Picture !

! .doc=Word.Document.8 !

! .pdf=AcroExch.Document !

.exc=txtfile

The **list with file extensions** with file type is readable by entering "assoc" without anything into CMD.

* **attrib:**

"attrib" changes the **attributes of files and folders**.

**+** = Adds an attribute

**-**  = Deletes an attribute

**R** = Read-only / protected file

**A** = Archivable file

**S** = Systemfile

**H** = Hidden

**/D** = Add/Delete the attributes also to folders

**/S** = Add/Delete the attributes for every file under the given folder

Examples:

attrib C:\WINDOWS\system32\taskmgr.exe **+H**

attrib C:\WINDOWS\system32\taskmgr.exe **+H +R –S**

# B

* **Break:**

This command determines if Ctrl+C is enabled or disabled. By default, the progress of a command or a batchfile stops when pushing Ctrl+C.

To pass this by, you can use BREAK OFF. If you want it to reactivate, you may use BREAK ON.

When having activated the command extensions, this command has no effect since Windows XP.

# C

* **cacls:**

This changes the settings of the "Access Control List (**ACL**) ". It adds or deletes the permission for a user to **access a file or a folder**. Therefore you need the other’s username and a connection to it to grant or to deny its permission to read, write or to change a file. The easiest way to remember:

/P Username:AccessOption ( Possible access options = N (No access)

= R (Read only)

= W (Write and read)

= C (Change, write and read)

= F (Full access))

Examples:

cacls "batch-Dictionary.txt" /p %username%:N

cacls "batch-Dictionary.txt" /p %username%:R

cacls "batch-Dictionary.txt" /p %username%:F

To pass by an answer of the user, you have to add the command "**echo J|**" in German or in English "**echo Y|**". You have to fit it onto your language for Yes.

**echo Y|**cacls "batch-Dictionary.txt" /p %username%:N

* **call:**

The call-command is used triple:

1. Call a file

2. **Call a batch-file**. Has the called file finished, the original one will continue.

3. **Call a spring point**.

To 1:

The command "call" should not be used to call a file, because the progress of the batch-file

**stops** until the program has finished. Instead you may use "***pushd, start, popd***".

To 2:

This is useful if you work with more than one file to make the current one a **Master-file**, the program also stops in this case, indeed.

To 3:

Therefore the call-command is really useful, because the program doesn’t end if the point does not exist. Instead, an error message will be displayed. The called point **stops his commands until** a **"*goto*-"** or an **"*exit* /b-"** command follows. In this case the batch-file **returns to the call-command** and **continues with the commands below**.

In addition you can use ***arguments***, but they are written with a ~ between the percent sign and the number. That means: **%~1, %~2** and so on:

call :Point1 firefox.exe "C:\Program Files\Mozilla Firefox"

pause

:Point1

echo %~1 %~2

exit /b

It would be displayed:

firefox.exe C:\Program Files\Mozilla Firefox

Because of the " " the path is just one argument, the " " are removed, though.

* **cd / chdir:**

This command **changes the directory** which the batch-file searches for commands apart from the paths given in ***path***. The command does not care about quotation marks.

**/D = Also changes the drive**

.. = Returns to the upper directory

Examples:

cd /D "C:\Program Files"

...is the same as...

cd /D C:\Program Files

cd .. If you are in C:\Program Files, you return to C:\

cd "C:\Documents and Settings\All Users\..\%username%\"

* This goes to "C:\Documents and Settings\%username%"
* **chkdsk:**

This command stands for "**ch**ec**k** **d**i**sk**". It analyzes a drive and displays its used and unused storage.

**/F** = Finds damaged sectors

**/R** = Recovers files in damaged sectors (requires /F)

Examples:

chkdsk F:

chkdsk F: /F /R

* **cls:**

Clears the screen of the batch-file. This does not affect its source code or other files!

* **cmd:**

Opens cmd.exe, the file that is usually used to open every batfile.

* **color:**

This command is used to **change the colors of background and font**. Counted in hexadecimal the **first** letter/number is the **background**, the **second** one the **font**.

0 = Black 8 = Dark grey

1 = Dark blue 9 = Blue

2 = Dark green A = Light green

3 = Dark turquoise B = Turquoise

4 = Dark red C = Red

5 = Purple D = Pink

6 = Dark yellow E = Yellow

7 = Grey F = White

Examples:

color 0e = Background black, font yellow.

color 1c = Background dark blue, font red.

color f4 = Background white, font dark red.

* **Command.com:**

Command.com is the older version of CMD.exe. This file is DOS-based and **cannot read names shorter than 8 symbols** (see also ***Shortnames***), it changes the drive differently (e.g. F:\ instead of cd /D F:\) and works slower. Command.com does not allow closing it by pressing the X at the bottom right corner but has to be terminated by entering ***EXIT***.

/P = Makes the program uncloseable, EXIT doesn’t work anymore. The process has to be killed.

* **comp:**

"comp" of "(to) **comp**are" compares to files whether they are identically or not.

Example:

comp "Path1\File1.txt" "Path1\File2.txt"

* **Compact:**

This command compresses or uncompresses files or entire directories. It has the following parameters:

/A = „Attributes“ = Displays hidden files and systemfiles.

/C = „Compress“ = Compresses the given file. If a folder is given, all added files in the

future will be compressed, too.

/i = „Ignore“ = Ignores errors. By default, the command stops on errors.

/F = „Force“ = Forces the process to compress the files, even if they have already

been compressed. These files are skipped by default.

/Q = „Quick“ = Queries only the most important information.

/S = „Sub“ = Executes the progress for all files in all subfolders.

/U = „Uncompress“ = Uncompresses the given file. If a folder is given, all added files in

the future will be (or stay) uncompressed, too.

Without parameters, the compression of the files from the current directory is displayed. It is also possible to enter in a single command multiple files or even placeholders.

Example:

Compact /C /I /Q "C:\Windows\Granit.bmp"

* **Convert:**

This command converts a partition from **FAT into NTFS**. Syntax:

CONVERT Partition [/V] [/X]

/V = Works more thorough, but takes also more time.

/X = Forces the partition to stop being ready to be used. All programs that are trying to

Access it are blocked.

Example:

Convert G: /X

* **copy:**

This command copies one file to another directory. The paths are written in quotation marks. The file can be renamed while being copied. You can also sum two files up to one. The following applies:

/b = Refers to a binary file (pictures, music, exe-files).

/y = No alert if a file shall be overwritten.

/-y = Extra alert if a file shall be overwritten.

Examples:

copy /B "C:\WINDOWS\Granit.bmp" "C:\Program Files\Granit.bmp"

copy "C:\Windows\Textdocument1.txt" "C:\Windows\Textdocument2.txt" "Texts.log"

It is also possible to allow the user to enter his/her own code which would be saved in a file. The command looks like that:

Copy con "Filename.Extension"

This command lets the user enter code until he or she presses Strg+Z. The entered code will be saved into the given file. Before using this command in a batch file, you may remember the user to enter Strg+Z at the end.

# D

* **date:**

The date is displayed and can be edited.

Example:

date 01.01.2000

* **defrag:**

This command defrags a drive. Extensions are possible:

**-a** = Analyze only

**-f** = Continue defragmentation despite errors

**-v** = Print more information

Example:

defrag C: -a

* **del / erase:**

These two commands are used to delete files (irreversibly) that have to be given with their entire path. Extensions improve the progress:

**/p** = Deletion has to be prompted by the user

**/f** = Protected files are deleted, too

**/s** = Every file and folder below the given one is deleted (see also ***Rd /Rmdir***)

**/q** = Filler texts like \* are allowed

Examples:

erase "C:\WINDOWS\Granit.bmp"

del "C:\WINDOWS\Granit.bmp"

erase /f /q "C:\WINDOWS\Granit.\*"

* **dir:**

**Lists all files** below the given path. If the path contains spaces, you have to use quotation marks.

**/A** = Attributes of the files: **D** = Folders ; **R** = Read only

**H** = Hidden ; **S** = Systemfile

**- before inverts their meaning.**

**/B** = Print only the file paths without any information.

**/C** = Displays the thousands separator (= 1.000 instead of 1000). This is normally

activated. /-C deactivates them.

**/L** = Use little letters only.

**/O:Order** = Order of the files: **N** = Name

**E** = Extension

**G** = Folders first

**D** = Date/Time (oldest first)

**S** = File size (smallest first)

**/P** = Pause after every full page.

**/Q** = Query the user.

**/S** = Display file and folder paths below the given one.

**/W** = Widescreen (File names side by side).

**/X** = Displays ***Shortnames***.

Examples:

dir "C:\Documents and Settings"

dir C:\

dir /S

dir /O:E /P C:\Programs

* **driverquery:**

Query the installed drivers. /V increases the printed information.

# E

* **echo. :**

The command "echo." makes a **blank** into the batch-window to make a text clearly.

Example:

@echo off

echo A nice day today.

echo.

echo And blanks are cool, too :D

pause

EXIT

* **echo:**

**1.** This command is used to **show text**. The annoying path that is shown every time you open CMD is deactivated by "**@echo off**". This is good for the overview.

Example:

@echo off

echo This message will be displayed.

pause

**2.** The command "echo" also allows you to edit, overwrite or to create files on the computer if writable with notepad. To create/overwrite a file, you use ">" after the line you would like to write. To edit a file, you use two of them ">>".

Examples:

echo I overwrite an existing file. > "C:\WINDOWS\Granit.bmp"

echo I create a new file. > "C:\WINDOWS\Neue Datei.txt"

echo I add a new line to an existing file. >> "C:\WINDOWS\kb900685.log"

* **edit:**

This command opens a **DOS-based editor** that can read faster than notepad. It contains a line- and column-system to make the reading easier and it has the same design as QBasic.

You can change colors or hide the mouse.

**/H** = Big, blue window like a Blue Screen.

**/B** = Small, black window like QBasic.

**/R** = Open a file in read-only mode.

Examples:

edit /H

This just opens the window.

edit /H /R "C:\WINDOWS\system32\shell32.dll"

* **endlocal:**

This command ends the area of "***setlocal***".

* **erase / del:**

These two commands are used to delete files (irreversibly) that have to be given with their entire path. Extensions improve the progress:

**/p** = Deletion has to be prompted by the user

**/f** = Protected files are deleted, too

**/s** = Every file and folder below the given one is deleted (see also ***Rd /Rmdir***)

**/q** = Filler texts like \* are allowed

Examples:

erase "C:\WINDOWS\Granit.bmp"

del "C:\WINDOWS\Granit.bmp"

erase /f /q "C:\WINDOWS\Granit.\*"

* **Errorlevel:**

An "errorlevel" is **reset after every command** if it didn’t work with the errorlevel. It is mostly used in ***IF***-commands, but it can do other things, too. The errorlevel is a number. Their meaning is often a secret, but many are steady:

Errorlevel 0 = Command was successfully executed without errors.

Errorlevel 1 = Command was successfully executed, but there were errors.

Errorlevel 9009= Command could not be found.

* **exit:**

This closes CMD.EXE.

**/B** = Just close the batch-file, but it lets the CMD-window opened.

* **explorer:**

Opens the Windows Explorer. By using "***start***" you can let Windows XP show alerts. Therefore you enter the following:

***start*** explorer "Path\File"

Usually there is an alert that tells you the owner could not be verified, but that’s often wrong.

# F

* **FC:**

Compares two files or two set of files and displays the differences.

Syntax for normal mode:

FC [/C] [/L] [/N] [/T] [/U] [/W] "Filepath1" "Filepath2"

Syntax for binary mode:

FC /B " Filepath1" " Filepath2"

/B = Compares in binary mode (e.g. pictures, Office Word-documents, exe-files etc.)

/C = Ignores case-sensitive

/L = Compares files in Ascii-mode (e.g. files from Notepad with codec ANSI)

/N = Displays the line numbers of the unique lines

/T = Does not change tabs to spaces

/U = Compares files in Unicode-mode (e.g. files from Notepad with codec Unicode)

/W = Compresses tabs and spaces

Examples:

FC /B "C:\Programs\Mozilla Firefox\firefox.exe" "%appdata%\Mozilla Firefox\firefox.exe"

FC /L /W "%userprofile%\Desktop\New Textdocument.txt" "C:\Programs\Batch\Test1.bat"

* **find:**

This command **searches one file for a particular string**. Then every found line will be shown. As heading the command prints the file name.

**/V** = Displays the lines that do NOT have the entered string.

**/C** = Displays the amount of the lines that contain the entered string.

**/N** = Displays the lines that contain the entered string with their line number.

**/I** = Ignores case-sensitive.

Example:

find /N "1" "New Textdocument.txt"

* **findstr:**

"findstr" is the more advanced command of "***find***". It finds a string in one or more files.

**/C:"String"**  = Looks for the entire string as such.

**/D:Directory** = Searches directories divided by semicolon.

**/I** = Ignores case-sensitive.

**/M** = **Displays only the file name(s)**.

**/N** = Prints the line number of every found line.

**/P** = Skips files with unprintable symbols.

**/S** = Looks for every file and folder below the entered file.

**/V** = Displays the lines that do NOT have the entered string.

**/X** = Displays the lines that are exactly the entered string.

Example:

findstr /C:"This program cannot be run in DOS mode." /M /S "C:\\*"

The disadvantage: If a file cannot be opened, an error message will be displayed. If there are too many lines that are too long for the command the batch-file crashes.

* **FOR:**

The FOR-command is often the **most important and the most flexible command**. Mostly the /F-extension is used to be as specific as possible. Here’s the syntax:

**FOR /F ["Options"] %%Variable IN (File-set) DO Command [Parameters]**

**FOR /F ["Options"] %%Variable IN ("String") DO Command [Parameters]**

**FOR /F ["Options"] %%Variable IN ('Command') DO Command [Parameters]**

or by using the option "usebackq":

**FOR /F ["Options"] %%Variable IN (File-set) DO Command [Parameters]**

**FOR /F ["Options"] %%Variable IN (' String ') DO Command [Parameters]**

**FOR /F ["Options"] %%Variable IN (`Command `) DO Command [Parameters]**

%%Variable is always **one letter**! Therefore you have to mind a is not the same as A.

The "Options" are as the following:

**eol=**c - Sets the **beginning of the line** for files (only one) that has to be ignored.

**skip=n** - Sets the amount of lines that have to be skipped at the beginning of a file.

**delims=**xxx - Sets the delimiters for many tokens. They replace the normal separators TAB

and space.

**tokens=**1,2-4, \* - Sets the amount of tokens that are used in the FOR-loop.

This allows more variables than the given one (%%var).

2-4 means 2,3,4. \* means every previous/following token

is set to one variable. For more about this watch below.

**usebackq** - Changes the syntax between the brackets. Usually useless.

The **variables** can only be one letter, but **after the** **"DO"-command** they are changeable:

%%~L - Removes every quotation mark of %%L.

%%~fL - Expands %%L to a **full file path**.

%%~dL - Generates only the **volume** of %%L.

%%~pL - Generates only the **path** of %%L without file name and volume.

%%~nL - Generates only the **file name** without extensionof%%L.

%%~xL - Generates only the **extension** of %%L.

%%~sL - Generated path contains only ***shortnames***.

%%~aL - Generates the **file attributes** of %%L.

%%~tL - Generates **date and time** of %%L.

%%~zL - Generates the **file size** of %%L.

%%~$PATH:L - Searches the ***PATH***-variable for the file and expands the first

find to a full file path. If none is found, %%L will be set to nothing.

These parameters can be combined:

%%~dpL - Generates the volume and the path of %%L.

%%~nxL - Generates the file name and the extension of %%L.

%%~fsL - Expands %%L to a full file path with ***shortnames***.

%%~dp$PATH:L – Searches the ***PATH***-variable for %%L and generates its volume and path.

%%~ftzaL - Expands %%L to a line similar to the ***DIR***-command.

Here’s an explanation of the options:

"**usebackq**" changes the **symbols between the brackets** between IN and DO that decide whether the words are files, strings or commands.

"**tokens**" and "**delims**" are easy to understand in this example:

FOR /F "tokens=1-5 delims=, " %%A IN ("This text will be shown. ") DO echo %%A %%B %%C %%D %%E

* "tokens" are from 1 to 5. "delims" are the **separators**: comma and space.
* In the brackets, there’s the string "This text will be shown." with a space between every word and in this case also a separator that was set by "delims". Because there are 5 words 1-5 and there are 5 "tokens", every word can be displayed.
* The first token is "This", because a space follows which was made by "delims" to a separator.

FOR /F "tokens=1,\* delims==" %%J IN ("Now the = is a separator, no comma or space anymore. ") DO echo %%J %%K %%L %%M

* Printed would be "Now the is a separator, no comma or space anymore. %L %M"
* **Token 1** is "Now the ", **token 2** is " a separator, no…anymore.", token 3 is not defined and token 4 as well not, so %L and %M are written.
* The **asterisk behind "tokens"** stands for **every following token**. As separator "=" was used, that’s why token 1 is "Now the ".

FOR /F "usebackq tokens=1,4 delims= " %%f IN ("Now another text is disassembled. ") DO echo %%f %%g **%%h**

* Printed would be: "Now is %h". Token 1 is "Now", token 2 is "another" etc, but **token 2,3,5… are not used**, but only 1 and 4, so only those are displayed. That’s why %%g is not token 2, but token 4.
* **%%h does not exist** as another token, because there are only 1 and 4 set to something. In this case "%h" is displayed as such.

Here are some examples of what the command can do:

FOR /F "delims=" %%A IN ('***dir*** /A /B /S "%homedrive%\"') DO (if "%%~**nx**A" == "firefox.exe" echo %%A)

=> Searches every path for "firefox.exe" and displays the found file.

FOR /F "tokens=1,2,\* delims=: " %%A IN ('***systeminfo***') DO (if /I "%%A" == "Operating system" echo You use %%B as operating system.)

=> This searches the output of "***systeminfo***" (not displayed) for "Operating system" and prints its value behind the colon. [Translation from German into English! "Operating system" perhaps is not written there. German word: "Betriebssystem".]

FOR /F %%A IN ('***tasklist*** /SVC') DO if /i "%%A" == "notepad.exe" ***taskkill*** /F /IM notepad.exe

=> Searches the output of “tasklist” and kills the process “notepad.exe” if found.

FOR /F "delims=" %%A IN ('***dir*** /A /B /-C /S') DO set /a Size=%Size% + %%~**z**A

echo The current folder is %Size% KB big.

=> Adds the file sizes of everything in the current folder and every subfolder and prints their sizes together.

You can do a lot more with the "FOR /F"-command, but there are also **/R** and **/L**:

**FOR /R [[Drive:]Path] %%Variable IN (File-set) DO Command [Parameters]**

With this command the program executes the command behind "DO" for every given file name in "(file-set) " under every folder entered behind /R. This allows for example to scan many files for their source code:

FOR /R "C:\" %%A IN (\*.txt) DO (

findstr /I /M /C:"This program cannot be run in DOS mode." "%%A" >nul

if "%ERRORLEVEL%" == "0" echo %%~**f**A cannot be run in Dos-mode.

)

=> This looks for the string behind /C: in every txt-file under C:\

**FOR /L %%Variable IN (Start,Step,End) DO Command [Parameters]**

This command executes the command after DO as often as the user wants it to. (Start,Step,End) is important in this point:

**The program steps in "Step" steps from "Start" to "End"** and executes the command until the end is reached:

FOR /L %%A IN **(1,1,9)** DO (if "%Place1%" == "%%A" set /a Place2=%%A +1)

=> The command is executed 9 times.

The FOR-command allows brackets to execute many commands in the loop.

FOR /L %%A IN (1,1,9) DO (

if "%Place1%" == "%%A" (

set /a Place2=%%A +1

))

**You must not forget to close the brackets! Otherwise the file crashes!**

**Overview syntax:**

FOR /F ["Options"] %%Variable IN (File-set) DO Command [Parameters]

FOR /F ["Options"] %%Variable IN ("String") DO Command [Parameters]

FOR /F ["Options"] %%Variable IN ('Command') DO Command [Parameters]

or by using the option "usebackq":

FOR /F ["Options"] %%Variable IN (File-set) DO Command [Parameters]

FOR /F ["Options"] %%Variable IN (' String ') DO Command [Parameters]

FOR /F ["Options"] %%Variable IN (`Command `) DO Command [Parameters]

FOR /R [[Drive:]Path] %%Variable IN (File-set) DO Command [Parameters]

FOR /L %%Variable IN (Start,Step,End) DO Command [Parameters]

* **format:**

Formats a drive.

/Q = Quick formatting (usually faster and also without errors)

* **FTP:**

„**F**ile **t**ransfer **p**rotocol“ to send and receive files over the Internet. This command it easy to understand, but you have to own an **FTP-capable server** with login to use this command to send or get files.

The FTP-command can either be used automatically or step by step.

Parameter for automatical version:

Syntax: **FTP** [-v] [-d] [-i] [-n] [-g] [-s:Filename] [-A] [Host]

-A Logs the user in as „Anonymous”.

-d Activates debugging.

-g Deaktivates placeholder.

-i Loads multiple files up without asking the user for each.

-n No automatical login demand after opening a website.

-s:*Datei* Enters a file with FTP-commands to read out.

-v Do not show information.

Host Webpage or IP.

Parameter for manual version:

? Queries all commands.

Ascii Sets the file transfer type to Ascii (.bat, .txt…).

Bell (De-)Activates an acustic sound when finished upload.

Binary Sets the file transfer type to binary (.exe, .zip…).

Bye / Quit Quits FTP.

Cd *Path* Changes the directory on the server.

Close / Disconnect Disconnects from the website.

Delete *File* Deletes a file from the server.

Dir Queries the directory on the server.

Get *File* Downloads a file.

Glob (De-)Activates placeholder for down- and upload.

Help *Command* Queries help to a particular command.

Lcd *Path* Changes the directory on the computer.

Mdelete *Files*  Deletes multiple files.

Mget *Files* Downloads mulitiple files.

Mkdir *Folder* Creats a folder on the server.

Mput *Files* Uploads multiple files.

O [www.*website*.de](http://www.website.de) O for „Open“. The website is never written with http://.

Prompt (De-)Activates automatical send modus.

Put *File* Uploads a file.

Pwd *Password* Sudden prompt after loginname.

Ren „*File“* „*File2“* Renames a file.

Rmdir *Folder* Deletes a folder on the server.

Verbose Queries more/less information on up- or download.

User *Username* Sudden prompt after opening a website.

In automatical version, the **commands from the manual** one are written **in**to **the file given behind –s**:.

Example:

ftp

o [www.batchlog.pytalhost.com](http://www.batchlog.pytalhost.com)

\*\*\*\*

\*\*\*\*

Ascii

Prompt

Lcd "%userprofile%\Desktop"

MPut "New Textdocument.txt" "New Skripttextdocument.npp"

Binary

Put Batch.html

Bye

* **ftype:**

This command **sets the program which opens a particular file type** (e.g. txtfile). It is often used under usage of "***assoc***" that allows to create an entire new file extension.

This requires a restart, though.

Syntax:

ftype Filetype=Commandline

Example:

ftype txtfile="%ProgramFiles%\Windows NT\Accessories\wordpad.exe" %1

# G

* **goto:**

This command allows you to go to a specific point in the batch-file, a ***springpoint***.

Examples:

:1

echo This Text is repeated endlessly.

goto :1

EXIT

goto :end

echo This text is not shown, because it is skipped.

:end

EXIT

* **gpresult:**

Lists some information about the computer settings. **/V** or **/Z** print more information.

# H

* **Help:**

This command queries some commands in Batch. In a few cases it can also be used to query help to a particular command: Help *Command*

# I

* **iexplore:**

Opens the Windows Internet Explorer. You can add a website to open behind:

iexplore [www.google.de](http://www.google.de)

* **if:**

The IF-command **sets conditions** which have to be complied to execute particular commands behind the IF-condition. The command allows brackets.

There are different types of the IF-command. Here’s an overview:

Type 1: IF [/I] [not] "%***Variable***%" comparison-operator "String" *command*

Type 2: IF [not] exist "Path" *command*

Type 3: IF [not] defined ***Variable*** *command*

Type 4: IF [not] ***errorlevel*** Number *command*

Comparison-operators:

**EQU** = **Equ**al

**NEQ** = **N**ot **Eq**ual

**GTR** = **G**rea**t**e**r** Than

**GEQ** = **G**reater and **Eq**ual

**LEQ** = **L**ess and **Eq**ual

**LSS** = **L**e**ss** than

**Comparison operators** are usually used to **work with numbers**. In this case, there should not be compared two strings by "==" in the line before or after.

**Type 1** of the IF-commands can be instead of "Variable==String" also be "Variable==Variable" or "String==String".

The following commands can be used with their entire extensions and parameters without limit. Thus, several conditions can be set:

If "%a%" == "1" (

If "%a%" == "2" (

If "%a%" == "3" ***echo*** a is 3.

))

You must not forget to **close the brackets**!

In work with numbers the comparison operators are used:

***Set*** /p Number=Enter here a number:

IF "%Number%" GTR "0" echo Your number was greater than 0.

IF "%Number%" LSS "0" echo Your number was less than 0.

IF "%Number%" EQU "0" echo Your number was exactly 0.

**Warning:** If one of the numbers has **more or less figures** than the other, the command causes an **error**!

Example: if "500" LSS "51" echo The command did something wrong!

To pass the mistake by, you can use the following command row:

***Set*** /p No1=No1:

***Set*** /p No2=No2:

***Set*** /a Difference=%Zahl1% - %Zahl2%

IF "%Difference:~0,1%" == "-" (***echo*** No1 is greater than No2.) ELSE (***echo*** No1 is less than No2.)

This is what it works like: It checks if the first figure of the difference of both numbers is a minus. If not, No2 has to be greater than No1.

The ***ECHO***-commands can be replaced by others.

The problem of this function is that both of the numbers must not be greater than “4.294.967.295” (~ 232 or 655362).

**Type 2** of the IF-commands may be the easiest one. You set the condition that if a file does [not] exist a specific command will be executed.

**Type 3** of the IF-commands refers to variables that are set by the ***SET***-command. If the variable is [not] set, the command behind will be executed:

If defined FilePath echo There is already set a file path!

**Type 4** of the IF-commands sets the condition that if the ***errorlevel*** is [not] equal a particular number the command behind will be executed.

If errorlevel 1 echo Error! The command could not be executed correctly!

# L

* **Label:**

The command “Label” creates, deletes or renames a partition. Syntax:

Label Partition:[Name]

Label [/MP] [Partition: | Partition-name]

Partition: = Example: “F:”

Partition-name = Example: “Daten“

/MP = Determines that the partition shall be held as mount point.

# M

* **Md / mkdir:**

**M**a**k**es a **dir**ectory (md=mkdir). It doesn’t matter which one of them you use.

Example:

mkdir "C:\Programs\New folder"

* **mode:**

This command changes settings of **devices** like the **window of CMD**. In this case the syntax looks as the following:

mode con CP SELECT=yyy

=> Changes the codepage (~Language)

mode con CP [/STATUS]

=> Displays the codepage.

mode con [COLS=c] [LINES=n]

=> Changes the window size: COLS=**Col**umn**s** / Width; LINES=Lines / Depth

=> At 1280x1024 **full screen is cols=157 lines=300**.

mode con [RATE=r DELAY=d]

=> Changes the speed of holding a button pressed (aaaaaaaaa…)

=> RATE=Speed of the repetition

=> DELAY=Duration until a button is repeated

* **more:**

Displays the source code of files paged.

**/E** = Activates the following extensions:

**/C** = Clears the screen before showing the next page

**/S** = Summarizes many blacks to one

**/Tn** = Changes tabs to *n* amount of spaces (Standard = 8)

**+n** = Starts with displaying the source code with the *n* th character

Example:

more /E /C /S "batch-Dictionary.txt"

Entering Q at --More-- ends the progress (Q of Quit).

* **move:**

**Moves files**. The path of the file to move and the path of the target have to stand in quotation marks. Between file and target has to be a space.

**/y** = No warning for overwriting a file.

**/-y** = Extra warning for overwriting a file.

Example:

move "C:\WINDOWS\Granit.bmp" "C:\Programs\Granit.bmp"

* **mplay32:**

This is the media player in C:\Windows\system32. It can only play a few formats and is usually not used.

Example:

***start*** mplay32 "C:\Documents and Settings\Username\My Documents\My Videos\Video.XXX"

* **Msg:**

This command is used up to "Windows XP Professional SP3". It **sends a message** to every on the current computer logged on user. A little window – usually in the middle of the screen – appears displaying a message.

Syntax:

Msg \* Message-Text

The star has to stay.

* **msgbox:**

This command is used since "Windows Vista" instead of "***Msg***". A little window – usually in the middle of the screen – appears displaying a message.

It’s the same Syntax **with star** here:

Msgbox \* Message

# N

* **net:**

If "net" works with **services**, the command uses the **displayname of services**.

**1.** net start "Service" : Starts an installed service.

**2.** net stop "Service" : Stops an installed service.

**3.** net pause "Service" : Pauses an installed, started service.

**4.** net continue "Service" : Continues a paused service.

**5.** net helpmsg *Number* : Displays the help message "Number".

**6.** net send *Message* : Sends a message to every home-connected computer.

**7.** net user: Creates, deletes or edits a user.

net user *Username* /ADD

net user *Username* /DELETE

net user *Username* Password|\*

* **netsh firewall add/delete allowedprogram:**

This command adds or deletes a program in the **Windows-Firewall** whether it is allowed to access the internet. The rule here:

"netsh firewall add allowedprogram Path\Filename **Filename** Mode"

or

"netsh firewall delete allowedprogram Path\Filename"

Examples:

netsh firewall add allowedprogram C:\WINDOWS\system32\cmd.exe **cmd.exe** ENABLE

netsh firewall delete allowedprogram C:\WINDOWS\system32\cmd.exe

* **netsh firewall set opmode:**

This command changes the **configuration of the Windows-Firewall** if there is no other installed firewall. Parameters are ENABLE and DISABLE. If there is no other firewall, there will be an alert if you use DISABLE saying the computer might be at risk.

Example:

netsh firewall set opmode disable

* **netsh firewall show opmode:**

This command shows the current configuration of the running firewall.

* **netstat:**

This command tells you your current connections to servers or other computers. There are a few extensions:

**-a** = Displays all connections and listening ports.

**-n** = Displays all connections in IPs.

# P

* **path:**

This command sets the **standard paths in** **which CMD looks for commands**. The paths before will be overwritten. Several paths are divided by semicolons.

* **pause:**

Pauses the progress of the batch-files and asks for pressing a key.

"***>nul***" suppresses the request "Press any key to continue . . .".

* **ping:**

This command sends a ping request to a website or an IP and prints the duration until an answer is given.

Example:

ping 65.99.250.115

* **ping localhost -n amount >nul:**

Pauses the progress of the batch-file for **n** pings (**1 second = 2 pings** ; in general:

**n+1 ping = n seconds**). Instead of "localhost" you can also use an IP.

* **Programpaths:**

%SystemRoot% = „Partition“:\Windows

%WinDir% = „Partition“:\Windows

%UserProfile% = „Partition“:\Documents and Settings\„User“

„Partition“:\Users\User

%AllUsersProfile% = „Partition“:\Documents and Settings\All Users

%AppData% = %UserProfile%\Application Data

%Temp% = %UserProfile%\Locale Settings\Temp

%SystemRoot%\Temp

**The following commands open windows:**

|  |  |
| --- | --- |
| Calc | Calculator |
| Certmgr.msc | Certification manager |
| Charmap | Character map |
| Cleanmgr | Clean-manager |
| Compmgmt.msc | Computer management |
| Cmd | Cmd.exe |
| Defmgmt.msc | Device management |
| Diskmgmt.msc | Disk manager |
| Drfg.msc | Defragmentation manager |
| Drwtsn32 | Doctor Watson, Windows debugger |
| Edit | DOS editor |
| Eudcedit | Character creation program |
| Eventvwr | NT Eventviewer |
| Start excel | Microsoft Office Excel |
| Explorer | Windows Explorer |
| Start firefox.exe | Firefox (Internet browser) |
| Freecell | Card game |
| Fsmgmt.msc | Folder security manager |
| Gpedit.msc | Group policies editor |
| iexplore | Internet Explorer (Internet browser) |
| Lusrmgr.msc | Local users manager |
| Magnify | Magnifier |
| Start moviemk | Movie Maker |
| Mplay32 | Media Player |
| Start mplayer2 | Media Player 2 |
| Start msconfig | Microsoft configuration program for startup and tools |
| Mshearts | Card game |
| Start msinfo32 | System information |
| Mspaint | Drawing program |
| Narrator | Microsoft Sam/Anna narrator |
| Notepad | Editor |
| Ntbackup | Backup manager |
| Ntmsmgr.msc | Removable storage manager |
| Ntmsoprq.msc | Removable storage operator requests |
| Start OneNote | Microsoft Office OneNote |
| Start Opera | Opera (Internet browser) |
| Osk | On-Screen-Keyboard |
| Perfmon.msc | Performance monitor |
| Start pinball | Pinball |
| Start powerpnt | Microsoft Office PowerPoint |
| Regedit | Registry-Editor |
| Regedt32 | Registry-Editor |
| Rsop.msc | Resultant set of policy connecting with the internet |
| Secpol.msc | Security policies |
| Services.msc | Services |
| Sndrec32 | Sound recorder |
| Sndvol32 | Volume, noises |
| Sol | Card game |
| Spider | Card game |
| Taskmgr | Task-Manager |
| Tourstart | Windows-Tour |
| Utilman | Program manager |
| Verifier | Driver validation assistant |
| Winhelp | Windows Help |
| Winmine | Game |
| Winver | Windows version |
| Start winword | Microsoft Office Word |
| Start wmplayer | Windows Media Player |
| Start wordpad | Editor |
| Write | Editor |

* **Prompt:**

This command changes the display when "echo on". Instead of "C:\Documents and Settings\User>" or "C:\Users\User" there is the thing you write behind prompt:

[

C:\Documents and Settings\User>prompt Hello all!

Hello all! echo Hi

Hi

Hello all! …

]

The prompt-command allows to write special symbols like "&" another way:

$A &

$B |

$C (

$D Current date

$E Escapesymbol (ASCII-Code 27)

$F )

$G >

$H Backspace (deletes the previous letter)

$L <

$N Current drive

$P Current drive and path

$Q =

$S (Space)

$T Current time

$V Windows 2000-Version

$\_ Blank

$$ $

* **pushd ; popd:**

PUSHD saves the current directory and changes to an entered one.

POPD returns to the folder that was saved by PUSHD.

Example:

echo %cd% Shows the current directory

PUSHD "C:\Programs\Mozilla Firefox" **Saves it** and **goes to Firefox**

start firefox.exe Starts Firefox out of "Mozilla Firefox"

popd **Returns to the saved folder**

# R

* **Rd/Rmdir:**

Removes a directory.

**/S** = Deletes every file and folder below the entered one.

**/Q** = No demand for /S.

* **reg:**

The command “reg” displays values or keys from the registry, changes or deletes them. Many main parameters use the following parameters:

Key = Registry path = Path to edit

/v Value = „value“ = Value inside the path to edit

/ve = „empty value“ = Edit the default value (Default) inside the path

/t Type = „type“ = Type of the value to edit

/d Data = „data“ = Data of the value to edit

/f = „force“ = Forces to overwrite the value or the key

To use the REG-command best, you should know a few tricks about the registry you could use. Some tricks can be viewed at the end of the explanation of the REG-command.

There are the following types (/t type) for the registry:

**REG\_SZ** = String

**REG\_DWORD** = Hexadecimal string

**REG\_BINARY** = Binary string

REG\_MULTI\_SZ = ???

REG\_EXPAND\_SZ = ???

REG\_DWORD\_BIG\_ENDIAN = ???

REG\_DWORD\_LITTLE\_ENDIAN = ???

REG\_NONE = ???

The first 3 types are the most commonly used ones.

Below the possible main parameters of the REG-commands:

- REG **ADD** Key [/v Value | /ve] [/t Type] [/d Data] [/f]

Adds a key or a value to the registry.

Example:

Reg add "HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\Virus.exe"

Reg add "HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\Virus.exe" /v Debugger /t REG\_SZ /d calc

**->** Creates a new key called “Virus.exe” which has the effect that instead of Virus.exe the calculator starts (calc).

- REG **COPY** Key1 Key2 [/s] [/f]

Copies all values (and under usage of /s also the subkeys) from Key1 into Key2.

Example:

Reg copy "HKCR\.bat" "HKCR\.command" /s

**->** Copies all keys and values from the file extension .bat to .command.

- REG **DELETE** Key [/v Value | /ve | /va] [/f]

Deletes a key, a single value or multiple values.

/va = „all values“ = Deletes all values in the key

Example:

Reg delete "HKCR\.command" /F

**->** Deletes the file extension .command if exist.

- REG **COMPARE** Key1 Key2 [/v Value | /ve] [/oa | /od | /os | /on] [/s]

Compares two keys or values..

/oa = Gives the differences and the similarities [a = all]

/od = Gives the differences only (default) [d = differences]

/os = Gives the similarities only [s = similarities]

/on = Gives neither differences nor similarities, just tells the user if similar or not

[n = none]

Example:

Reg compare "HKCR\.bat" "HKCR\.cmd"

- REG **EXPORT** Key File

Exports an entire key into a .REG-file which restores the key on double click automatically. Used to make a backup or manual system restores.

Example:

Reg export "HKCR\batfile" "%userprofile%\Desktop\batfile.reg"

**->** Makes a backup of the key "batfile".

- REG **IMPORT** File

Imports a .REG-file which can restore a key. The file has to be saved by REG EXPORT before.

Exmaple:

Reg import "%userprofile%\Desktop\batfile.reg"

**->** Restores the keys and values from "batfile.reg"

- REG **QUERY** Key [/v Value | /ve] [/s]:

Searches the registry for the give path or value and outputs them.

Example:

Reg query "HKLM\Software\Microsoft\Windows\CurrentVersion\Run" /S

**->** Queries the startup key of all users.

- REG **RESTORE** Key File

Restores a key from the given file into the given key. The given file has to be saved by REG SAVE before.

Example:

Reg restore "HKCR\cmdfile" "%userprofile%\Desktop\cmdfile.reg"

**->** Restores the key "HKCR\cmdfile" from the file "cmdfile.reg".

- REG **SAVE** Key File

Saves the given key into the given file to allow a restore by REG RESTORE. The file will be saved in an unreadable format. This way, data privacy is better granted than by REG EXPORT.

Example:

Reg save "HKCR\cmdfile" "%userprofile%\Desktop\cmdfile.reg"

**->** Makes a backup of the key "HKCRcmdfile" into the file "cmdfile.reg".

Tips and tricks you can use in the registry:

1. Edit group policies via batch:

"HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced"

/v Hidden /t REG\_DWORD /d 1 **=** Show hidden files

/v HideFileExt /t REG\_DWORD /d 0 **=** Show file extensions

/v HideIcons /t REG\_DWORD /d 0 **=** Show icons

/v ShowSuperHidden /t REG\_DWORD /d 1 **=** Show system files

/v DisableThumbnailCache /t REG\_DWORD /d 1 **=** Block Thumbs.db

/v SuperHidden /t REG\_DWORD /d 1

**=** System files can be hidden better than normally hidden files

/v StartMenuRun /t REG\_DWORD /d 0

**=** Remove "Run" from the start menu

/v Start\_ShowRecentDocs /t REG\_DWORD /d 0

**=** Remove "Recent Documents" from the start menu

"HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer"

/v SearchHidden /t REG\_DWORD /d 1

**=** Include hidden files on Windows-Search

/v SearchSystemDirs /t REG\_DWORD /d 1

**=** Include system files on Windows-Search

"HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer"

/v NoSimpleStartMenu /t REG\_DWORD /d 1 **=** Force simple start menu

/v NoRun /t REG\_DWORD /d 1 **=** Forbid "Run"

/v NoChangeStartMenu /t REG\_DWORD /d 1

**=** Forbid changing the settings of the start menu

/v NoChangeTaskBar /t REG\_DWORD /d 1

**=** Forbid changing the settings of the taskbar

/v NoClose /t REG\_DWORD /d 1

**=** Remove "Shutdown" from the start menu

/v NoStartMenuMorePrograms /t REG\_DWORD /d 1

**=** Remove "All programs" from the start menu

/v NoStartMenuMFUprogramsList /t REG\_DWORD /d 1

**=** Remove the list of most frequently used programs from start menu

/v NoFind /t REG\_DWORD /d 1

**=** Remove "Search" from the start menu

/v StartMenuLogOff /t REG\_DWORD /d 1

**=** Remove "Logoff" from the start menu

/v LockTaskbar /t REG\_DWORD /d 1

**=** Force fixing the taskbar

/v HideClock /t REG\_DWORD /d 1

**=** Remove the Windows-clock from the taskbar

/v NoTrayContextMenu /t REG\_DWORD /d 1

**=** Forbid context menu from the taskbar (right click)

/v NoFolderOptions /t REG\_DWORD /d 1

**=** Forbid changing the folder options via control panel

/v NoFileMenu /t REG\_DWORD /d 1

**=** Remove "File"-context menu from the explorer (upper left corner)

/v NoDesktop /t REG\_DWORD /d 1

**=** Hide desktop symbols (can be found in the folder)

/v NoPropertiesMyComputer /t REG\_DWORD /d 1

**=** Hide the properties of my Computer

/v DisableRegistryTools /t REG\_DWORD /d 1

**=** Forbid editing tools for the registry (e.g. regedit etc.)

/v ForceClassicControlPanel /t REG\_DWORD /d 1

**=** Force usage of classic control panel

"HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\System"

/v DisableTaskMgr /t REG\_DWORD /d 1

**=** Disable Windows Task-Manager

/v NoDispSettingsPage /t REG\_DWORD /d 1

**=** Forbid changing the display settings

/v NoDispScrSavPage /t REG\_DWORD /d 1

**=** Forbid changing the screen saver settings

/v NoDispAppearancePage /t REG\_DWORD /d 1

**=** Forbid changing the colors

/v NoDispBackgroundPage /t REG\_DWORD /d 1

**=** Forbid changing the desktop background

/v NoDispCPL /t REG\_DWORD /d 1

**=** Hide control panel

"HKLM\Software\Microsoft\Windows\CurrentVersion\policies\system"

/v HideStartupScripts /t REG\_DWORD /d 0

**=** Show the startup scripts of Windows

/v HideShutdownScripts /t REG\_DWORD /d 0

**=** Show the shutdown scripts of Windows

1. Various:

"HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options" is an important registry key that is often infected by viruses.

It can be **used to prevent** particular **processes from running** or to redirect a process to another one. This can be done as the following:

1. **Add a new key** to the registry key from above. The key is **called the same as** the name of **the process to prevent** from running (e.g. setup\_wm.exe).
2. **Add a** **new value to this subkey** from type string (REG\_SZ) **called** “**Debugger**”.
3. Add to the value "Debugger" the process name that should be run instead as data. **If none is given, nothing happens when executing the process.**

Startup keys:

* "HKCU\Software\Microsoft\Windows\CurrentVersion\Run"
* "HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce"
* "HKLM\Software\Microsoft\Windows\CurrentVersion\Run"
* "HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnce"
* **REM:**

This command is used to help those who read the source code or for yourself to give information about for example the syntax of a command or ideas for following functions.

This command does not do anything when executing.

REM This will not be shown while executing the file, but it still exists.

* **Ren:**

Renames a file: REN "Filepath1" "Filepath2"

* **Runas:**

**Replaces “Run as…”.**  With this command, you can determine under which user a program is run. Syntax:

RUNAS /user:*Username* "*Program name or path to a file*"

“User*name”* has to be replaced by the wanted user.

“*Program name or path to a file*” has to be replaced by the wanted program. Make sure to put an entire path in if the program is not placed in system32/system64! Otherwise, the command might cause errors!

When having run the command, **the program asks for a password**. This password is **not viewed** in CMD, so it can neither be read, nor copied into a file. This way, **it is protected**.

If either password or username are incorrect, the program outputs an error message. This **error message cannot be suppressed by** using "***2>nul***", **but by** "***>nul***". If you use "***>nul***", though, the dialogue asking for a password won’t be viewed, too. To pass this by, you can use "***echo*** Please enter the password: ".

# S

* **sc (~net):**

This command, similar to the ***NET***-command, only works with **services**.

It uses their **true service names** in contrast to "***NET***". Syntax in general:

**sc [Command] [Service name] [Parameters]**

**query** Displays the **status** of a service.

**queryex** Displays the **status** of a service detailed.

**start** Starts a service.

**pause** Pauses a service.

**continue** Continues a paused service.

**stop** Stops a service.

**config** Configures a service:

**start**= <boot|system|auto|demand|disabled>

--> Changes the **start configuration**.

--> boot = BEFORE Windows.

--> system = While Windows startup screen.

--> auto = While logging on.

--> demand = Has to be started manually.

--> disabled = Cannot be started.

**error**= <normal|severe|critical|ignore>

--> Changes the **configuration for errors**.

--> normal = Error is logged, user will be warned.

--> severe = **Computer restarts** with the last known, good configuration.

--> critical = **Computer restarts** with the last known, good configuration.

If that doesn’t work, there will be a **Blue Screen**.

--> ignore = Error will be ignored.

**depend**= Service-name(s)

--> Determines which service(s) has/have to be started before this one.

**Several service names are divided by "/".**

**DisplayName**= "Display-name"

--> Configures the **display name of the service**.

**--> The space behind "=" must not be removed!**

**description** **Changes** the description of a service.

**failure** Changes the command if a service crashes.

--> **reset** = Duration of running the service before crash

--> **reboot** = Message to be displayed before restart

--> **command** = Command line that has to be executed if the service crashes.

**qc** Displays **information of the configuration** of a service.

**qdescription** **Displays** the service’s description.

**qfailure** Displays the action that is taken if the service crashes.

**delete** Deletes the service in the registry.

**create** Creates a service (also in registry).

**GetDisplayName** Displays the **display name of a service**.

**GetKeyName** Displays the **true name** of a service (**works with display names**).

**EnumDepend** Lists the services that cannot be started without running the entered one.

These commands do not need a service name:

**boot** (ok|bad) Determines whether the last configuration was good or bad.

Lock Locks he service database.

QueryLock Displays the lock status.

Examples:

Sc config Schedule start= demand

Sc start Schedule

AT 20:00 ***msg*** \* hi

***REM*** Schedule = Task scheduler

* Sets the start mode of the service to manual, starts it and runs a command.

----

Sc stop Themes

Sc config Themes start= disabled

***REM*** Themes = Designs

* Stops the service and sets the start mode to deactivated, so that it cannot be restarted. This forces to use the Windows 98 design.
* **set:**

This command is used to set ***variables***. It allows mathematically calculations or user inputs:

**/a** = Variable works with numbers

**/p** = Waits for a **user prompt**

Examples:

set fire=C:\Programs\Mozilla Firefox\firefox.exe

set /a Calculation=5 + 5

set /p Password=Please enter the password:

set /p Number=Enter one of the following figures: 1,2,3

set /a Number=%Number%

---> If the user enters a letter, **"Number"** will be set by the "SET /A"-command to **0.**

"set" can also be used to **replace a string inside a variable** with another string. This means that a letter, a figure or a word can be removed and replaced by something else. Syntax:

Set Variable2=%Variable1:TextToRemove=TextToInsert%

It is possible to **replace the existing variable** (1) instead of creating another one (2).

Example:

Set stones=This is an example for replacing parts of a text.

Echo % stones %

Set stones =% stones:parts=things%

Echo % stones %

* First " This is an example for replacing parts of a text." is shown,

then " This is an example for replacing things of a text.".

The word "parts" was replaced by "things".

* **setlocal:**

"setlocal" alone creates a **room inside the batch-file** in which ***variables*** that are set before the command are ignored. This can be imagined like that:

[ BATCH-FILE ] -> The room of the batch file is limited by [ ].

[ BATCH- **(Setlocal)** FILE ] -> Setlocal creates a new room **( )** in the batch file [ ].

The command "setlocal" is ended by "***endlocal***".

Extensions:

setlocal enabledelayedexpansion / disabledelayedexpansion:

"enable" allows to call ***variables*** per **!var!**. Else than ***%var%*** this version also works inside brackets ( (Command1 Command 2 Command 3...) ).

Example:

setlocal enabledelayedexpansion

set Number=0 REM The variable "Number" is set to 0.

if "%Number%" == "0" ( REM By using "IF" a bracket is opened.

set Number=%Number% +1 REM "Number" is set +1 inside them.

echo !Number! is 1. REM The number INSIDE the brackets is printed.

echo %Number% is 0. REM The number OUTSIDE the brackets is printed,

) REM because CMD looks for the variable in general

REM and starts searching the batchfile outside the brackets.

setlocal enableextensions / disableextensions:

This command activates or deactivates the command extensions in CMD. This can prevent your file from executing commands you don’t want it to or to have many possibilities. Usually they are activated.

The "setlocal"-command has no effect outside of batch-files.

* **shift:**

This command changes the **position of *arguments*** (%1, %2... %9). This allows using **more than 10 arguments** at the same time. If you want to keep the first arguments, you can use **/N.** The command will **skip the first arguments** till the N’th one.

shift **/2** = %0 and %1 will be skipped, %3 goes to %2, %8 to %7 etc.

* **Shortnames:**

Shortnames are **shortened names of files and folders** which are **longer than 8 characters**. They used to be usually used in DOS, because it couldn’t read names that are longer than 8 characters. They also **never contain spaces**.

Shortnames are easy to find out: You take the first 6 characters and add a **~1**.

If necessary the **file extension has to be added** behind the **~1**.

Following this principle "batch-Dictionary.docx" will be "BATCH-**~1**.**docx**".

If there are **several files or folders with the same first 6 characters**, you have to watch the alphabet.

The second file / folder becomes from **~1** to **~2**, the third one to **~3** etc.

For example this path:

C:\Documents and Settings\Username\Local Settings\Temp\batch-Dictionary.docx

C:\DOCUME~1\USERNA~1\LOCALS~1\TEMP\BATCH-~1.DOCX

Shortnames are important in ***Command.com*** above all.

* **shutdown:**

The shutdown-command is used to shut down Windows, to restart or to log off:

**/a** = Abort shutdown

**/c** = Add a **comment** (followed by "Comment" with " ")

**/f** = Force Windows to end the processes

**/l** = Log off

**/r** = Restart

**/s** = Shutdown

**/t** = **Time** in seconds till **shutdown** (followed by a number)

Example:

Shutdown –s –t 120 –c „Windows will shut down in 120 seconds.“

* **sndrec32:**

Opens the Windows Soundrecorder. It can be used to play a .WAV-file automatically:

Start sndrec32 “Path to the file.WAV” /play

To let it close after having played the wavefile, add /close:

Start sndrec32 “Path to the file.WAV” /play /close

* **sort:**

This command sorts the source code of a readable file or a text.

* **Springpoints:**

Spring-points are points inside the batch-file that can be navigated by the ***goto***- or the ***call***-command. They consist of a **colon and a word** and/or a number. There can be used ***arguments*** for them.

Here’s an example for a spring-point:

:Name\_Of\_The\_Point

Spring-points **never contain spaces**. Otherwise only the first word is its name.

* **start:**

The start-command is an important command to **start programs without stopping** the progress of the batch-file.

The file to open has to be **WITHOUT " "**, because otherwise another CMD window will open with the title between the " ".

To pass this by, you can use the commands "***pushd***" or "***cd/chdir***". If there is even a space in the filename, you have to use its shortname (see "***Shortnames***").

You can also add **/D** (= **/D**irectory) behind *start*, to enter the path to the file in quotation marks.

The command allows entering the process‘ priority:

/LOW

/BELOWNORMAL

/NORMAL

/ABOVENORMAL

/HIGH

/REALTIME

But also the window mode:

/MIN

/MAX

Unlike the ***CALL***-command another batch-file will not be loaded into the current one, but start in a new CMD window which allows running two batch-files at the same time.

Examples:

start /MIN ***sndrec32*** /play /close "%SystemRoot%\Media\ding.wav"

start /LOW ***iexplore*** <http://www.google.com>

start **/D** "%userprofile%\Desktop" *Filename*

* **Symbols (& / || / ² / ...):**

Symbols are often a problem because they are printed else in batch than in notepad. Here are some:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* Explanation: \*

\* -How to place a file into the German startup folder: \*

\* \*

\* The file that has to create another file has to be written by an active CMD window (Start- \*

\* Run-cmd), so that the Ü of Startmenü can be recognized. In the source code the Ü will be \*

\* shown as an unknown square which must not be changed. \*

\* \*

\* The source code together with the square can be copied and pasted into another place. \*

\* Here are some characters that are not recognized in batch: \*

\* \*

\* This is an ü (small)...:  \*

\* This is an Ü (big)....: š \*

\* This is a ß...........: á \*

\* This is an ä (small)...: „ \*

\* This is an Ä (big)....: Ž \*

\* This is an ö (small)...: ” \*

\* This is an Ö (big)....: ™ \*

\* This is an alert: - cannot be written in Office Word, sorry - \*

\* This is a filled space: Û \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

But these are not the only one that has to be passed by:

& = Command separation. Divides command1 and command2 in one line.

&& = Command separation. Command2 is only executed if command1 was successful.

| = Command separation. Divides command1 and command2 as &, but it can be used

another way too. For more watch below.

|| = Command separation. Command2 is only executed if command1 was not successful.

% = Used for ***Variables***.

( = Opens a sub command chain.

) = Ends a sub command chain.

How to pass by:

Often you can place a ^ before the symbol. This allows showing it in echo for example.

%-characters have to be replaced with a double %: %%.

Usage of |:

Writing the normal syntax

"**echo input|Command**"

the demand of "command" will be **automatically answered by** "input". Therefore it is important if there is a **space** between "input" and |. Here’s an example:

**echo Y|**cacls "C:\Programs\Mozilla Firefox" /p %username%:N

--> Because there is the question "**Are you sure? (Y / N)**" in "***cacls*** ..." and there will be automatically filled in **"Y" because of the echo-command,** you can change the access without the user’s validation.

The same applies to other commands like e.g. "**help|sort**". In this command the output of "help" will be sorted by alphabet.

**For more symbols see also *Ascii***.

* **systeminfo:**

Displays much information about the operating system.

# T

* **taskkill:**

This command is the more advanced type of "***tskill***" and also works instead of XP Home on Vista. It has a few important extensions:

**/F** = **Forces to end** the process.

**/IM** = **Name** of the process with extension.

**/PID** = **Process-ID**.

**/T** = Ends the whole **process structure** (=also subordinated ones).

You can use /IM and /PID several times in one command.

Examples:

taskkill /IM explorer.exe

taskkill /F /IM notepad.exe /PID 3056 **REM notepad.exe may not be PID 3056!**

* **tasklist:**

This command lists all **running processes** with the following information:

Processname, PID, Session Name, Session ID, Memory.

**/FI "Filter"** = Filter to use.

**/FO Format** = Possible formats are "table", "list" and "csv".

**/M Module** = Shows every task with its loaded modules as (DLL-files).

**/NH** = Leaves the **heading away** for the formats table and csv.

**/V** = Shows detailed information:

status, username, runtime (doesn’t work!), window title

**Filter:**

Filter name Possible operators Possible values

--------------- ------------------------- ------------------

STATUS eq, ne RUNNING | NOT RESPONDING

IMAGENAME eq, ne Processname

PID eq, ne, gt, lt, ge, le PID-value

SESSION eq, ne, gt, lt, ge, le Session ID

SESSIONNAME eq, ne Session name

CPUTIME eq, ne, gt, lt, ge, le CPU-runtime in:

hh:mm:ss.

hh - hours,

mm - minutes, ss - seconds

MEMUSAGE eq, ne, gt, lt, ge, le Memory usage in KB

USERNAME eq, ne Username

SERVICES eq, ne Service name(s)

WINDOWTITLE eq, ne Window title

MODULES eq, ne DLL-Name(s)

eq = **Eq**ual

ne = **N**ot **E**qual

gt = **G**reater **T**han

ge = **G**reater and **E**qual

le = **L**ess and **E**qual

lt = **L**ess **T**han

/FI "STATUS eq NOT RESPONDING"

* **time:**

Prints the **current time** and asks for a change.

**/T** only prints the time.

"time *Time*" changes the time to "*Time*", e.g. time 20:00 .

* **title:**

Changes the window title: title "Title"

* **tree:**

**Prints the directory as a tree** of the current folder or the entered one.

**/A** = Uses ***Ascii***-characters instead of advanced ones.

**/F** = Also prints **file names**.

"Path" can be used to enter a directory whose structure shall be shown.

Example:

tree /F "C:\"

* **tskill:**

This is a pretty old command that is used up to Windows XP SP-3. The better command is "***taskkill***" with more extensions.

Here, the process name has to be used without file extension:

tskill notepad

tskill 3056

* **type:**

Shows the content of a text-based file: ' type "File path" '

Another command for this function is ***more***.

# U

* **user&user32:**

These commands are **DLL**-files that can be used with "**rundll32.exe**". They allow you to **change system- und device settings** from e.g. mouse or keyboard:

rundll32 user32.dll,SwapMouseButton

rundll32 user.dll mouse, enable

rundll32 user.dll keyboard, enable

# V

* **Variables:**

Variables are values that are set by the ***SET***-command. Usually they are called by enclosing the variable into **% %**.

Under usage of "***SETLOCAL*** ENABLEDELAYEDEXPANSION" they can be called within brackets by enclosing them into **! !** (see also ***setlocal***).

set Variable1=echo This is one

set Stone=of the most important things in CMD.

%Variable1% %Stone%

The output would be "This is one of the most important things in CMD." by the echo-command.

Variables are a bit easier than ***arguments***, but they follow the same principle.

The following variables are usually already set:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| %ALLUSERSPROFILE% | | | C:\Documents and Settings\All Users | |
| %APPDATA% | | C:\Documents and Settings\!User!\Application Data | | |
| %CD% | | | !Current directory! | |
| %CmdCmdLine% | | | "C:\WINDOWS\system32\cmd.exe" | |
| %CommonProgramFiles% | | | C:\Programs\Common Files | |
| %COMPUTERNAME% | | | !COMPUTERNAME! | |
| %ComSpec% | | | C:\WINDOWS\system32\cmd.exe | |
| %date% | | | !Current date! | |
| %errorlevel% | | | !Errorlevel of the previous command! | |
| %FP\_NO\_HOST\_CHECK% | | | NO | |
| %HOMEDRIVE% | | | C: | |
| %HOMEPATH% | | | \Documents and Settings\!USER! | |
| %LOGONSERVER% | | | \\!COMPUTERNAME! | |
| %NUMBER\_OF\_PROCESSORS% | | | !AMOUNT OF PROCESSORS! | |
| %OS% | | | Windows\_NT | |
| %Path% | C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem | | | |
| %PATHEXT% | | | .COM;.EXE;.BAT;.CMD | |
| %PROCESSOR\_ARCHITECTURE% | | | ! PROCESSOR-CONSTRUCTION! | |
| %PROCESSOR\_IDENTIFIER% | | | ! PROCESSOR\_INFOS! | |
| %PROCESSOR\_LEVEL% | | | ! PROCESSOR-LEVEL! | |
| %PROCESSOR\_REVISION% | | | ! PROCESSOR-???! | |
| %ProgramFiles% | | | C:\Programs | |
| %PROMPT% | | | $P$G | |
| %random% | | | Random number between 1 and 65536 | |
| %SESSIONNAME% | | | Console | |
| %SystemDrive% | | | C: | |
| %SystemRoot% | | | C:\WINDOWS | |
| %TEMP% | | | | C:\DOCUME~1\!USERNAME!\LOCALS~1\Temp |
| %time% | | | | !Current date! |
| %TMP% | | | | C:\DOCUME~1\!USERNAME!\LOCALS~1\Temp |
| %tvdumpflags% | | | | ? |
| %USERDOMAIN% | | | | !COMPUTERNAME! |
| %USERNAME% | | | | !USERNAME! |
| %USERPROFILE% | | | | C:\Documents and Settings\!USERNAME! |
| %windir% | | | | C:\WINDOWS |

* **ver:**

Shows the current Windows version.

# W

* **wmic:**

This complicated command often has to be **installed** before being available. Yet, you can do a lot with it. For a list of what it handles with, use "wmic /?".

To receive help to a specific handle from wmic, use "wmic Handle /?" and so on.

For nearly every command you can use one of the followings:

ASSOC

CALL

CREATE

DELETE

GET

LIST

SET

Useful commands are:

Wmic **startup** get Description,Command

* Shows programs that are run at startup.

Wmic **process** get ExecutablePath,ProcessId

* Shows the currently running processes with their path and PID.

Wmic **service** get DisplayName,Name,PathName,ProcessID

* Shows services with display names, true names, process path and PID of the currently running process if available.

**For further information like syntaxes and commands read the intern help.**

# X

* **xcopy:**

This command is the same as ***copy***, but it is a lot more specific:

**/C** = Continues copying **despite errors**.

**/E** = Copies empty folder, too, **but no not-empty folders**! (useful with /S)

**/F** = **Prints the names** of source path and target path.

**/G** = Allows copying encrypted files into a place which does not support encryption.

**/F** = Copies files with the **attributes** **'hidden'** and **'system'**.

**/K** = Copies **every attribute**. Usually 'read-only' is deleted.

**/L** = Lists the currently copied files.

**/O** = Copies **information like user and ACLs** (Access Control List).

**/P** = Demands for every file to copy.

**/Q** = Does not show any information.

**/R** = Overwrites read-only files.

**/S** = Copies every file and folder that is not empty.

**/X** = Copies **file audit settings** (requires /O).

**/Y** = No demand for overwriting a file.

**/-Y** = Extra demand for overwriting a file.

**/Z** = Copies files in a mode that allows a restart.

Difference:

copy "C:\Programs\Mozilla Firefox" "C:\Software\Mozilla Firefox"

copies files, but not the folders below.

xcopy /s "C:\Programme\Mozilla Firefox" "C:\Software\Mozilla Firefox“

also copies the folders.

Another example:

xcopy "C:\WINDOWS\" "F:\WINDOWS\" /S /E /L /G /H /R /K /O /X /-Y

# Command order for beginners

|  |  |
| --- | --- |
| 1. Echo | 40. Path |
| 1. Pause | 41. Type |
| 1. Cls | 42. Sort |
| 1. Color | 43. More |
| 1. Exit | 44. Find |
| 1. Title | 45. Findstr |
| 1. Ping localhost | 46. Xcopy |
| 1. >nul | 47. *Errorlevel* |
| 1. Ping | 48. Netsh |
| 1. Shutdown | 49. Netstat |
| 1. Cd / chdir | 50. *Arguments* |
| 1. Chkdsk | 51. %0 |
| 1. Comp | 52. Shift |
| 1. Copy | 53. Defrag |
| 1. Move | 54. Format |
| 1. Date | 55. Attrib |
| 1. Time | 56. Assoc |
| 1. Del / Erase | 57. Ftype |
| 1. Ver | 58. Reg |
| 1. Edit | 59. *Symbols* |
| 1. Start | 60. Cacls |
| 1. *Program paths* | 61. User / User32 |
| 1. Call | 62. Systeminfo |
| 1. Goto | 63. Gpresult |
| 1. *Spring points* | 64. Driverquery |
| 1. Rem | 65. Tasklist |
| 1. Ren | 66. Tskill |
| 1. Dir | 67. Taskkill |
| 1. Tree | 68. Pushd / Popd |
| 1. Cmd | 69. Mode |
| 1. Explorer | 70. Setlocal |
| 1. Command.com | 71. Endlocal |
| 1. Md /mkdir | 72. Net |
| 1. Rd / rmdir | 73. Sc |
| 1. Set | 74. Wmic |
| 1. *Variables* | 75. *Shortnames* |
| 1. %var% | 76. For |
| 1. %var:~x,y% | 77. *Ascii* |
| 1. If |  |

Play around with the command, if possible, before reading the new one. Have fun batching!

For every complaints and questions contact 414-702-600 in ICQ or GrellesLicht28 in YouTube.