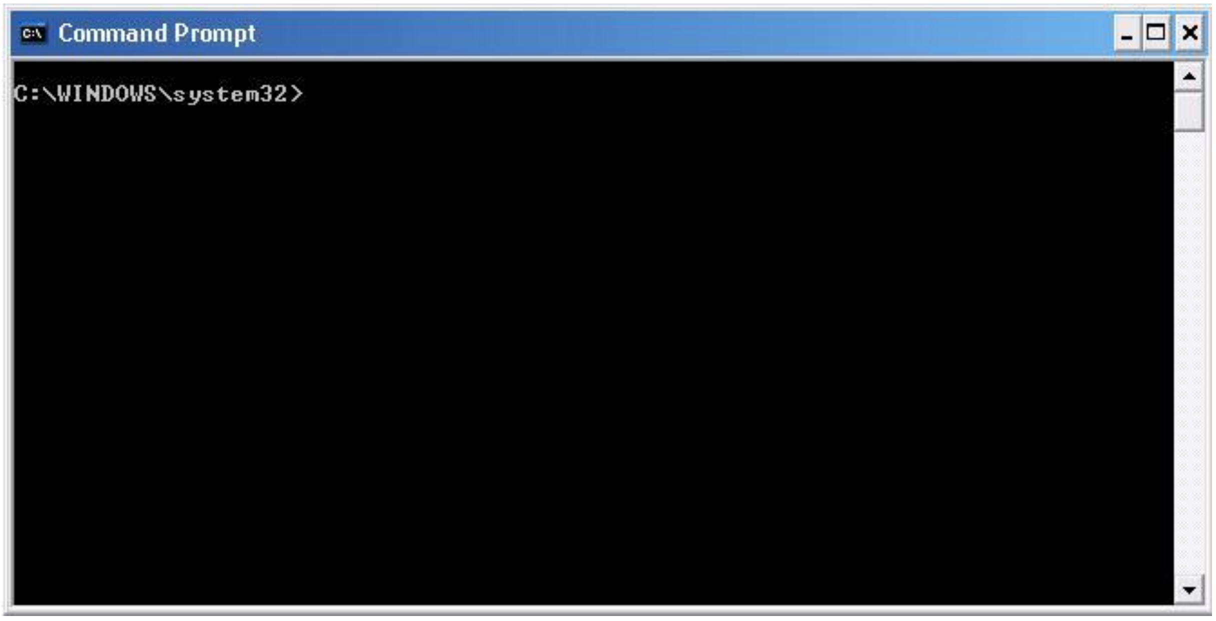
Batch-Dictionary

Overview:

%var%	Color	Explorer	Md / Mkdir	Rem	Tree
%var:~x,y%	Command.com	Extrac32	Mode	Ren	Tskill
%var:a=b%	Comp	FC	More	RunAs	Type
%0	Compact	Find	Move	Sc	User/User32
>nul	Convert	Findstr	MPlay32	Set	*Variables*
>"Path"	Copy	For	Msg	Setlocal	Ver
2>&1	Date	Format	Msgbox	Shift	Wmic
Arguments	Debug	FSUtil	Net	*Shortnames*	XCopy
Ascii	Defrag	Ftp	Netsh	Shutdown	
Assoc	Del / Erase	FType	Netstat	Sndrec32	
Attrib	Dir	Goto	Path	Sort	
Break	Driverquery	GPResult	Pause	Start	
Cacls	Echo	Help	Ping	*Symbols*	
Call	Edit	IExplore	*Program-paths*	Systeminfo	
Cd / Chdir	Endlocal	If	Prompt	Taskkill	
Chkdsk	Erase / Del	Label	Pushd / Popd	Tasklist	
Cls	*Errorlevel*	Labels	Rd / Rmdir	Time	
Cmd	Exit	MakeCab	Reg	Title	

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



```
C:\WINDOWS\system32>
```

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 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 ***label*** 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**" 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.

To suppress the output of an entire batch file, you can use the following command row:

```
@echo off
If "%1" == "NextStep" goto :Start
Cmd /c %0 NextStep >nul
Exit /b
:Start
[Commands]
```

First the batch file checks if ***argument***1 is "*NextStep*". If so, it jumps to :Start. If not, it restarts CMD with itself (***%0***) using "NextStep" as argument1. ">nul" suppresses the output of the command "cmd /c %0 *NextStep* >nul". As it calls the batch file itself, though, the output of the batch file will be suppressed.
The same applies for the case you want to suppress errors. You just need to change the ">nul" to "2>nul".

- **>"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"

As 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 ***labels***:

```
@echo off
set Variable=do a lot.
call :Label3 With this you can "%Variable%"
pause >nul
:Label3
echo %1 %2 %3 %4 %5
exit /b
```

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*

Hold "Alt" and push the numbers of the following table on the **numeric keypad**:

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 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} (Line Feed)	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} (Carriage Return)	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: →	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: Đ	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: ▄	

{CR} can be saved into a variable and used via !CR! (under usage of ***Setlocal*** EnableDelayedExpansion) by using the following command:
FOR /F %%A IN ('***copy*** /Z "%~dpf0" nul') DO ***set*** "CR=%%A"

{LF} can be saved into a variable and used via !LF! (under usage of Setlocal EnableDelayedExpansion) by using "set LF=^" and leaving two empty lines below.

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

- <u>**attrib:**</u>

"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

<u>Examples:</u>

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 batch file 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**, but the program also stops in this case.

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 **executes its 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.

- **cd / chdir:**

This command **changes the directory** that 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 "**check disk**". 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. The command has the following parameters:

/C	= Executes the command or the file and closes.
/K	= Executes the command or the file and does not close.
/Q	= Executes "@echo off".
/T:XY	= Determines background and font colors. More information in "***color***".
/E:On	= Enables command extensions.
/E:Off	= Disables command extensions.
/V:On	= Executes the command "***setlocal*** EnableDelayedExpansion".
/V:Off	= Executes the command "***setlocal*** DisableDelayedExpansion".

- **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.

Additionally, you have the opportunity to show text in multiple different colors. To do so, you can use this function:

```
:ColorText [%1 = Color] [%2 = Text]
set /p ".=." > "%~2" <nul
findstr /v /a:%1 /R "^$" "%~2" nul 2>nul
set /p ".={BS}{BS}" <nul
if "%3" == "end" set /p ".=  " <nul
del "%~2" >nul 2>nul
exit /b
```

The "{BS}{BS}" **have to be replaced** by two *Ascii*-Codes 08!
The function is called by this command:

```
CALL :ColorText [Color] [Text] ["end"]
```

At first, a file with the namen "Text" is created without content. Here, you should keep in mind that you can only use symbols that are allowed in filenames. Thus, **quotation marks are not valid**. Furthermore, it is possible that files with the wanted text as names already exist. If so, the old one will be overwritten. This can be prevented by **creating and switching** into an entirely **new folder**:

```
MD UniqueFolderName
CD UniqueFolderName
```

After the file has been created, it will be queried by the FINDSTR-command in the color the user wants it to. Afterwards, the file will be deleted. The IF-command checks if the current function call is supposed to be the last one.

Here an example source code of a whole file:

```
////////////////////////////////////
@echo off
MD UniqueFolderName
CD UniqueFolderName
Echo Colored text here:
Echo.
Call :ColorText 0B "This"
set /p ".=  " <nul
Call :ColorText 0C "is"
set /p ".=  " <nul
Call :ColorText 01 "cool" end
Echo.
Echo.
pause
CD..
RD /S /Q UniqueFolderName
exit
```

```
:ColorText [%1 = Color] [%2 = Text]
set /p ".=." > "%~2" <nul
findstr /v /a:%1 /R "^$" "%~2" nul 2>nul
set /p ".={BS}{BS}" <nul
if "%3" == "end" set /p ".=  " <nul
del "%~2" >nul 2>nul
exit /b
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
```

To have text between the separate colored parts you can use
set /p ".= " <nul
The last color call has an "end" as third ***argument***.

- ## 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 does not 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 decompresses files or entire directories. It has the following parameters:

/A	= "Attributes"	= Displays hidden files and system files.
/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”	= Decompresses 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 that 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 Ctrl+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 Ctrl+Z at the end.

D

- **date:**

The date is displayed and can be edited.

Example:
date 01.01.2000

- **debug:**

The DEBUG-command works on a **16-bit-base** and thus under the process ntvdm.exe. Unfortunately, **64-bit-systems do not own this file** anymore and the **command loses any effect**. It works with assembler-code. Therefore, it has a huge amount of possibilities. Since this dictionary predominantly works with batch, there are only a few tricks below.
Before, though, the data processing system needs to be explained.

Every single place, column and line, **has** a fix **hexadecimal value**. Consequently, the upper left corner has the value 0000. As it would take too long to list every single place, here a short **formula to calculate it automatically**:

```
Set /a Conv=(%Y% * 80 + %X%) * 2
Cmd /c exit /b %Conv%
Set HexCode=%=ExitCode:~4%
```

X is the **column** and **Y** the **line**. You start counting with "0" at both:
0 <= X < 80
0 <= Y < 50

1st: **Show symbols in any place in the CMD-window**
It is possible to show any (printable) symbol in any place in CMD in any color ("*color*").
Because of the 16-bit-base, it **only works for 80 columns and 50 lines** (for more details see "*mode*"). Anything further will be cut out.
Syntax:

```
(echo EB800:%HexCode% "%Symbol1%"%Color1% "%Symbol2%"%Color2% […]
Echo Q) | debug >nul
```

Symbol1, symbol2 etc. will be written in a row. **To write on separate positions**, you need to **use multiple ECHO-commands** with different %HexCode%. This is especially useful if you know the code even before the file execution.

Example:

```
(echo EB800:01E0 "c"0a "o"0a "o"0a "l"0a
Echo EB800:00A0 "T"0c "h"0c "a"0c "t"0c "Û"00 "i"0e "s"0e
Echo Q) | debug >nul
```

2nd: **Activate and use the mouse**

To allow the usage of the mouse is not always as comfortable for the user as it seems to be in the beginning because he/she can only access the places that are checked by the program. Since the code is multiply used in this case, it is recommended to **create a file called "mouse.dat"**.
Here the common code:

```
:: Create file (only once)
Echo e100 B8 1 0'3'DB CD'3'B0 3'C'CD'3'B DB't'FA 91 D3 EA D3 E8 91 C3>mouse.dat
Echo g116>>mouse.dat
Echo q>>mouse.dat
:: Execute file
FOR /F "tokens=6-8 delims== " %%A in ('Debug ^< mouse.dat ^| Find /i "X"') do (
:: Set X- and Y-value
Set /a X=0x%%A
Set /a Y=0x%%C
:: Calculation of the clicked place
Set /a Conv=!Y! * 160 + !X! * 2
Cmd /c exit /b !Conv!
Set HexCode=!=ExitCode:~4!
)
:: Check for particular places
If "!HexCode!" == "…" [Commands]
```

This code requires "Setlocal EnableDelayedExpansion". Executed once, the FOR-command is no problem. If it is **used in** a **loop**, though, so that the user has the opportunity to click infinitely with the mouse, it **uses** the **CPU to full capacity** (1 core).
As the **code** has to be **exact** as above **up to the end of the FOR-loop**, here an example down from the IF-commands:

```
If "!HexCode!" == "01E0" goto :Start
If "!HexCode!" == "01E2" goto :Start
If "!HexCode!" == "01E4" goto :Start
If "!HexCode!" == "01E6" goto :Start
If "!HexCode!" == "01E8" goto :Start
If "!HexCode!" == "008C" exit

If "!HexCode!" == "0090" exit
If "!HexCode!" == "0092" exit
```

Goto :ExecuteMouse

To understand how the FOR-command works in this case, it is recommended to debug "mouse.dat" with the DEBUG-command (without >nul) and to click anywhere with the mouse afterwards. Then it becomes obvious how the X- and the Y-value are found out.

- **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
```

To write two different text passages into one single line you need to use the following script:

```
SET /P ".=Enter Text No. 1 here" <NUL
SET /P ".=Enter Text No. 2 here" <NUL
```

This way, two texts are printed into one line. The same applies for any amount of texts, numbers, string etc. The only important thing, though, is not to use " .

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 did not 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.

- **Extrac32:**

Decompresses a cabinet file (.CAB-file) that contains one single file.
Syntax: EXTRAC32 "Path of the .CAB-File" "Name of the extracted file"

Example:
Extrac32 "***%userprofile%***\Desktop\Test.cab" "Important document.txt"

This command can be well combined with the command "***makecab***".

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"

Using a combination of several commands, it is possible to **find out** the **target file's path of a shortcut**:

For /F "skip=3 delims=" %%A IN ('**find** ":\" "*[Path to the shortcut]*"') DO ***Echo*** %%A

⇨ By this FOR-command, the first 3 lines of the output of the FIND-command are skipped and **only the linked path** will be displayed. This is made possible by the fact that each shortcut contains the path to its target file in text format.
The 3 skipped lines are a blank, the path to the shortcut and the drive of the target.

The problem with this method, though, is that the **parameters/*arguments*** that are used for executing the target file **are removed**.
To view them as well, the ***MORE***-command has to be used in combination with ***SET***-commands, with whom you define an array, taking only the parameter. Since this would be too complicated and too long, further explanation is omitted in this case.

- **findstr:**

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

/A:ColorCode	= Sets the colors used in case of requiring line numbers or file names A list of usable colors is available in the ***COLOR***-section.
/C:"String"	= Looks for the entire string as such.
/D:Directory	= Searches directories divided by semicolon.
/F:"File"	= Reads a list of files to search from the file “File”
/G:"File"	= Reads the string to look for from the file “File”
/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 contain the entered string.
/X	= Displays the lines that are exactly the entered string.

Examples:
```
findstr /M /S /C:"This program cannot be run in DOS mode." "C:\*"
findstr /A:0a /S /G:"String.log" "*.txt"
```

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.

"findstr" as ***ECHO***-command:
*This way is very inaccurate. It is recommended to use the function in "**color**".*
The color-function of "findstr" allows using multiple colors at the same time. In combination with the ***SET***-command as ECHO-command, it is possible to determine the place of these colors. The command row below is the best way to do this.

```
Set /p ".=Text before text to mark" <nul                 | Write text
MD UniqueTempFolder                                      | Create new folder
CD UniqueTempFolder                                      | Switch to new folder
Echo Text after text to mark>"Text to mark later on"     | Create a file
Findstr /A:Color /S /C:"Text after text to mark" "*"     | Query file colored
CD..                                                     | Switch to upper directory
RD /S /Q UniqueTempFolder                                | Remove created folder
```

The first output of the file is "*Text **before** text to mark*" in the SET-command. After that, it creates a non-existing folder. It changes the directory to this folder. Inside, it creates a file called "*Text to **mark** later on*" with the content "*Text **after** text to mark*". Then the FINDSTR-command finds in the file "*Text to **mark** later on*" the "*Text **after** text to mark*".
After the output, it returns to the upper directory and removes the created folder again.

<u>The output is:</u>

```
Text before Text to mark: Text after
```

It is also possible to have multiple colors in one line by using two FINDSTR-commands. To pass the word wrap by after the first FINDSTR-command, it is necessary to replace the single ECHO-command with a SET-command:

```
Set /p ".=Text before anything" <nul                              | Write text
MD UniqueTempFolder                                               | Create new folder
CD UniqueTempFolder                                               | Switch to new folder
Set /p ".=Text after first marked" <nul >"First text to mark "    | Create 1st file
Set /p ".=Text after second marked" <nul >"Second text to mark"   | Create 2nd file
Findstr /A:Color /S /C:"Text after first marked" "*"              | Query 1st file colored
Findstr /A:Color /S /C:"Text after second marked" "*"             | Query 2nd file colored
CD..                                                              | Switch to upper dir.
RD /S /Q UniqueTempFolder                                         | Remove created folder
```

<u>The output is:</u>

```
Text before First marked Text after first Second marked Text after second
```

The first three parts stayed the same. There are only two new parts in the output: The second file and the content of the second file.

There are a few disadvantages of this command row. First there is a colon behind every marked (=colored) text. Second, it creates files. Third, it will be messed up in case there are multiple files containing a particular text. Fourth, it will cause errors if it has not enough rights to create folders or files.

• 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 is 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 <u>one</u> 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 extension of %%L.
%%~xL	- Generates only the **extension** of %%L.
%%~sL	- Generated path contains only ***<u>shortnames</u>***.
%%~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 is 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 is 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 is 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 "%%~nxA" == "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% + %%~zA
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 %%~fA 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)

- **FSUtil:**

This command stands for **F**ile **S**ystem **Util**ity and consequently works with everything around files. In the following, there is first the syntax of a particular part and then its function.

FSUtil File CreateNew ["File path"] [File size in byte]
- Creates a file in the given path with a certain size. Per byte, the file contains a "{NUL}".

FSUtil File SetShortname ["File path"] ***Shortname***
- Determines the 8-point-3-names (Shortnames) of a file. This can be useful for the command "***Start***". The short name does not need to have an extension.

FSUtil FSInfo Drives
- **Lists all installed drives and partitions**. More information see the end of this command.

FSUtil FSInfo DriveType [Drive or partition]
- Outputs the **type of drive** (e.g. CD-ROM-Drive; Implemented drive). Drives have to be in the format C:.

FSUtil FSInfo VolumeInfo [Drive or partition]
- Outputs some **informationen about the drive**, e.g. the file system. This does only work **if** the **device is ready**. CD-ROM-drives without inserted CD, for example, occur an error. Drives have to be in format C:\.

FSUtil Volume DiskFree [Partition]
- Outputs the **maximal available and** the **free disk space of a partition**. Partitions have to be in format C:.

As it can be useful to **query the type of each drive**, here a command row:

Windows XP:

```
Setlocal EnableDelayedExpansion
fsutil fsinfo drives>FSUtil.tmp
set counter=0
FOR /F "delims=" %%A IN ('more FSUtil.tmp') DO (
        set /a counter+=1
        set Partition!counter!=%%A
)
FOR /L %%A IN (1,1,!counter!) DO fsutil fsinfo drivetype !Partition%%A:~-3!
```

```
Del /F FSUtil.tmp >nul
```

Windows Vista and 7:

```
FOR /F "delims=" %%A IN ('fsutil fsinfo drives') DO call :DriveType %%A
pause
exit
:DriveType
if "%1"=="" exit /b
fsutil fsinfo drivetype %1
shift
goto :DriveType
```

- **FTP:**

Uses the "**f**ile **t**ransfer **p**rotocol" to send and receive files over the Internet. This command is 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	Deactivates placeholder.
-i	Allows to load up multiple files.
-n	No automatical login demand after opening a website.
-s:*File*	Enters a file with FTP-commands to read out.
-v	Do not show information.
Host	Website or IP.

Parameter for manual version:

!	Switches to CMD. Return back by using "*exit*".
?	Queries all commands.
Ascii	Sets the file transfer type to Ascii (.bat, .txt…).
Bell	(De-)Activates an acoustic 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*	Creates a folder on the server.
Mput *Files*	Uploads multiple files (requires –i).
O www.*website*.com	O for „Open“. The website is never written with http://.
Prompt	(De-)Activates automatical send modus.
Put *File*	Uploads a file.
Pwd	Queries the current remote directory.
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 the automatical version, the **commands from the manual** one are written **in**to **the file given behind –s**:.

Example:

```
ftp
o www.batchlog.pytalhost.com
****
****
Ascii
Prompt
Lcd "%userprofile%\Desktop"
MPut "New Textdocument.txt" "New Scripttextdocument.npp"
Binary
Put Batch.html
Bye
```

If you do not want to create a file, for example because you do not have access to the hard drive or because the login data can be read out, you can alternatively use ***Echo***-commands:

```
(
        Echo o www.batchlog.pytalhost.com
        Echo user [Username] [Password]
        Echo Prompt
        Echo Dir
        Echo bye
) | ftp –n
```

This way, everything will be written directly into FTP and executed.

- **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 ***label***.

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

- **if:**

The IF-command **sets conditions** that 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 is 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 = **Gr**eat**er** Than
GEQ = **G**reater and **Eq**ual
LEQ = **L**ess and **Equ**al
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" (~ 2^{32} or 65536^{2}).

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

- **Labels:**

Labels 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 is an example for a label:

```
:Name_Of_The_Point
```

Labels **never contain spaces**. Otherwise, only the first word is its name.

M

- ## MakeCab:

This command compresses a single file into a cabinet file (.CAB-file).
Syntax: MAKECAB [/Vn] [/L "Directory"] "File to compress" "Cabinet file.CAB"

/Vn	= **Verbosity.** "n" is a number between 1 – 3. "3" is the strongest compression.
/L "Path"	= Path to the folder into which the .CAB-file shall be saved.

Examples:
Makecab "***%userprofile%***\Desktop\New textdocument.txt" "Old textfile.cab"
Makecab /V3 "Batchfile.bat" "Batchfile.cab"
Makecab /L "%userprofile%\Downloads" "Picture.jpg" "Photo.cab"

This command can be well combined with "***EXTRAC32***".

- ## Md / mkdir:

Makes a **dir**ectory (md=mkdir). It does not 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**.
=> At 1920x1080, **full screen is cols=237 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 small 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 is 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 **default 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. The command has the following parameters:

-T	= Sends an infinite amount of pings. Ctrl+C stops the progress.
-N n	= n is the amount of pings to send (Default: 4).
-L n	= n is the amount of bytes to send per ping (Default: 32).
-W n	= n is the duration in milliseconds until the next ping is sent at last (Default: 5000)

In the HOSTS-file (X:\Windows\system32\drivers\etc\hosts), there can be some redirections. Often the IP of the current session (127.0.0.1) is associated with the word "localhost". Usually there is only this single redirection.

To make a time-bound pause in a batch file you can use
"ping localhost –n Time-in-seconds+1 ***>nul***".

Examples:
```
ping –T 65.99.250.115
ping localhost –N 5 >nul
ping 209.85.135.105 –W 1001 –n 2 >nul
```

- **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:

Adress book	Wab (Start)
Audio settings	Mmsys.cpl
Backups	Ntbackup
Calculator	Calc
Card game	Freecell
Card game	Mshearts
Card game	Sol
Card game	Spider
Certificate manager	Certmgr.msc
Character builder	Eudcedit
Character map	Charmap
Clip board (Cache)	Clipbrd
Cmd.exe	***Cmd***
Computer Management	Compmgmt.msc
Control Panel	Control
Creator of self-extracting files	Iexpress
Defragmentation	Dfrg.msc
Desktop Settings	Control desktop
Device Manager	Devmgmt.msc
DirectX Diagnostics	Dxdiag
Disk Cleanup Wizard	Cleanmgr
Disk Management	Diskmgmt.msc
Doctor Watson, Windows Debugger	Drwtsn32
DOS text editor	***Edit***
Driver checker wizard	Verifier
Event logging / Event viewer	Eventvwr
Firefox (Internet browser)	Firefox (Start)
Folder security / Sharing folders	Fsmgmt.msc
Folder Settings	Control folders

Font Settings	Control fonts
Game	Winmine
Group policy editor	Gpedit.msc
Group Policy evaluation on the Internet	Rsop.msc
Help and Support	Helpctr
Import utility for adress book	Wabmig (Start)
Internet Explorer (Internetbrowser)	***IExplore*** (Start)
Internet Settings	InetCPL.cpl
Keyboard Settings	Control keyboard
Language assistant	Narrator
Locale security policies	Secpol.msc
Magnifier	Magnify
Manager of removable storage	Ntmsmgr.msc
Manager of removable storage	Ntmsoprq.msc
Media Player 2	Mplayer2 (Start)
Media Player	***Mplay32***
Microsoft configuration program about startup and tools	Msconfig (Start)
Microsoft Office Excel	Excel (Start)
Microsoft Office OneNote	OneNote (Start)
Microsoft Office PowerPoint	Powerpnt (Start)
Microsoft Office Word	Winword (Start)
Mouse settings	Control mouse
Movie Maker	Moviemk (Start)
Netmeeting	Conf (Start)
Network Connections	Control netconnections
Objekt packager	Packager
ODBC-Data source-Administrator	Odbccp32.cpl
“Locale users and groups” – manager	Lusrmgr.msc
On-Screen-Keyboard	Osk
Opera (Internet browser)	Opera (Start)
Outlook Express	Msimn (Start)
Paintbrush	MsPaint
Performance monitor of the system	Perfmon.msc
Phone Dialer	Dialer (Start)
Pinball	Pinball (Start)
Power Management	Powercfg.cpl
Printers and faxes	Control printers
Regional and Language Options	Intl.cpl
Registry editor	Regedit
Registry editor	Regedt32
Remote Desktop	Mstsc
Scanner and camera settings	StiCPL.cpl
Security center	Wscui.cpl
Services	Services.msc
Soundrecorder	***Sndrec32***
SQL Server Client Network Utility	Cliconfg
System configurations editor	Sysedit

System information	Msinfo32 (Start)
System Settings	Sysdm.cpl
Task-Manager	Taskmgr
Task Scheduler	Control schedtasks
Telephone link manufacturer	Telephon
Telnet Utility	Hypertrm (Start)
Text editor	Notepad
Text editor	Wordpad (Start)
Text editor	Write
Utility manager	Utilman
Volume and sounds	Sndvol32
Windows-Tour	Tourstart
Windows chat utility	Winchat
Windows Explorer	***Explorer***
Windows help	Winhelp
Windows Media Player	Wmplayer (Start)
Windows version	Winver

- **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 writing special symbols like "&" another way:

$A	&
$B	|
$C	(
$D	Current date
$E	Escape symbol / Backspace (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
$$	$

- **<u>pushd ; popd:</u>**

PUSHD saves the current directory and changes to an entered one.
POPD returns to the folder that was saved by PUSHD.

<u>Example:</u>

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.

<u>Example:</u>

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.

<u>Example:</u>

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

<u>Example:</u>

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]

<u>Example:</u>

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.

<u>Example:</u>

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.

<u>Exmaple:</u>
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.

<u>Example:</u>
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.

<u>Example:</u>
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.

<u>Example:</u>
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. <u>Edit group policies via batch:</u>

```
"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

2. 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"

Make the ***START***-command easier:

In the key "HKLM\Software\Microsoft\Windows\CurrentVersion\App Paths", you can assign a path to an executable file. When the user uses "Start>Run" or the search of Windows Vista or Win7, the system searches this key for the executable file. If it exists, it takes its path and starts the file there.
This is how to create a new rule:

1. Right click onto "App Paths" > New key. The new key has to be the name of the executable.
2. In the right window, you need to change "Default" to the full path of the file, e.g. "C:\Programs\My own program\Start.exe".
3. Furthermore, you need to add a new string called "Path". Its value needs to be the same as "(Default)", but without \Start.exe.

<u>Example:</u>

1. App Paths > New key > "Minecraft.exe"
2. (Default) > "C:\Documents and Settings\User\Application Data\.minecraft\bin\Minecraft.exe"
3. New > String > "Path" > "C:\Documents and Settings\User\Application Data\.minecraft\bin"

<u>Find out the codepage of a system:</u>

The codepage determines the order of the characters in the Ascii-table. "ACP" is the codepage for the language of the system; "OEMCP" is the default one in CMD.

```
reg query "HKLM\SYSTEM\CurrentControlSet\Control\Nls\CodePage" /v "ACP" ^| find /i "ACP"
reg query "HKLM\SYSTEM\CurrentControlSet\Control\Nls\CodePage" /v "OEMCP" ^| find /i "OEMCP"
```

<u>Useful folders:</u>

```
reg query "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders"
```

<u>Influence Windows-Sounds:</u>

You can determine in which case which sound (.WAV-file) is played. A list of all possibilities can be viewed by using the following command:

```
reg query "HKCU\AppEvents\Schemes\Apps\.Default"
```

<u>To change:</u>

Reg add "HKCU\AppEvents\Schemes\Apps\.Default\*[Event]*\\.Current" /ve /d "..\\..\\*[Path on %SystemDrive% without drive]*" /F

<u>Example:</u>

```
Reg add "HKCU\AppEvents\Schemes\Apps\.Default\Minimize\.Current" /ve /d "..\..\Windows\Media\notify.wav" /F
```

- **REM:**

This command is used to help those who read the source code or for oneself 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.

Additional options for "query" and "queryex":
type= <driver|service|all>
--> **Type of services** to display (default: service).
--> driver = Drivers
--> service = Services
--> all = Drivers & Services

state= <inactive|active|all>
--> **State of services** to display (default: active).
--> inactive = Disabled, currently not running
--> active = Enabled, currently running
--> all = Enabled and disabled services

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

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".

Another function of the SET-command is explained in "***ECHO***". It is used to print multiple unique texts in one single line.

- **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*** via **!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 /a 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 batch file 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 do not 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** until the Nth 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 as so called "8-dot-3 names". This means that the names are never longer than 8 letters, a dot plus 3 letters for the file extension. 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**.**DOC**".
Notice that the file name consists of exactly eight letters (including ~1) and the file extension only of three letters.

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.DOC

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

An easy method to find out the shortname of a particular file is the following:

FOR /F "delims=" %%A IN ("Filename") DO ***set*** Shortname=%%~sA

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 wave-file, 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.

- **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
```

In addition, you can start another CMD-window (or a batch file) in the current one by using "start **/B**". This allows **multithreading**. All windows or files work at the same time. The new instance or the file runs in its **own process**. **Variables** of the one are **not set in the other**, so there is no actual communication between the threads. It is possible to pass this by by using extra files, though:
You **write** the **text** you would like to send to the other process **into a file** and **let the other** one **read** it **out**.

Example:

```
Start /B SecondFile.bat
```

Often it is **best to start** the **current file** another time. **If** this is **made without further ado**, the file will be started infinitely which may **lead to a system crash**. The following code is used to get two processes out of one file:

```
@echo off
IF "%1" == "SecondProcess" goto :SecondProcess
Ping localhost –n 2 >nul
Start /B %~nxs0 SecondProcess
:FirstProcess
[Insert commands here]
Exit
: SecondProcess
[Insert commands here]
Exit
```

Each thread / Each file / Each instance has to be closed by an own EXIT-command.

- **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 is 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 an 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- and 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***%	PG
%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
2. Pause	41. Type
3. Cls	42. Sort
4. Color	43. More
5. Exit	44. Find
6. Title	45. Findstr
7. Ping localhost	46. Xcopy
8. >nul	47. *Errorlevel*
9. Ping	48. Netsh
10. Shutdown	49. Netstat
11. Cd / chdir	50. *Arguments*
12. Chkdsk	51. %0
13. Comp	52. Shift
14. Copy	53. Defrag
15. Move	54. Format
16. Date	55. Attrib
17. Time	56. Assoc
18. Del / Erase	57. Ftype
19. Ver	58. Reg
20. Edit	59. *Symbols*
21. Start	60. Cacls
22. *Program paths*	61. User / User32
23. Call	62. Systeminfo
24. Goto	63. Gpresult
25. *Spring points*	64. Driverquery
26. Rem	65. Tasklist
27. Ren	66. Tskill
28. Dir	67. Taskkill
29. Tree	68. Pushd / Popd
30. Cmd	69. Mode
31. Explorer	70. Setlocal
32. Command.com	71. Endlocal
33. Md /mkdir	72. Net
34. Rd / rmdir	73. Sc
35. Set	74. Wmic
36. *Variables*	75. *Shortnames*
37. %var%	76. For
38. %var:~x,y%	77. *Ascii*
39. 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.

Contributors:

GrellesLicht28	-	Almost everything
GeniusAlex	-	Codes in ***debug***
tDwtp / DDlol01	-	Windows7-function of ***FSUtil***

Sources:

- Windows XP Professional SP3 – CMD-Help sections
- Windows XP Professional SP3 – Help- and Support center
- www.dostips.com
- www.source-center.de
- www.youtube.com