REINIER MALIEPAARD

Windows Command Line

C:\Users\You> MOVE /Y "knowledge" "skills" [ENTER]

Table of contents

1. Why an ebook on the Windows Command Line?

Since I’m a Linux user for some years, I love working with the Linux Command Line (1) and for good reason: many administrative tasks can be done with ease. While also working on Windows, I’ve to admit that I use the Command Line less often. However, I discovered that the Windows Command Line with its simple syntax offers a lot of interesting ways to automate monotonous and repetitive tasks. In Linux, I use the command line for basic actions like

- copy, rename, move, or delete files
- create, rename or delete (sub)directories
- navigate from one directory to another

The same actions could be done rather easily with the Windows Command Line.

2. Make an example directory

To learn the most of this ebook, you should make an example directory ‘test’ on a drive (in my case drive D: but any other will work as well of course) and a subdirectory ‘my Doc’. Put then some files into these directories. For now, you can do this your own way, without using the Windows Command Line. So select a drive and create

a. the directory ‘test’
b. the subdirectory ‘test\my Doc’ (with a space in the subdirectory name)

and put

c. some (empty) files into the directory ‘test’
d. and some files into ‘test\my Doc’

My D:\test contains the following eight files: ‘file1.txt’, ‘file2.dat’, ‘file2.txt’, ‘file3.txt’, ‘pic001.jpg’, ‘pic002.jpg’, ‘musicBach.mp3’ and ‘musicBrahms.mp3’.

I placed two files ‘doc 1.rtf’ (with space!) and ‘doc2.rtf’ (without space) into D:\test\my Doc.

Remember that the words ‘current directory’ in the following text refer to D:\test.

3. A little exercise: open the Windows Command Prompt window and go to the example directory.

Open the Windows Command Prompt window by following these steps:

a. Click Start
b. Type in the Search (or just start typing): CMD and press Enter.

You should now see the Windows Command Prompt window, a window with the so called prompt like C:\> (see 3.1). To be more precise: invoking CMD (better known as CMD.exe) opened this Windows Command Prompt window.

But there is more: the commands you'll type at the prompt, will be interpreted by the same CMD.exe. So, remember CMD.exe as the Windows Command (line) Interpreter, as a type of Windows shell.

3.1 The prompt

The Windows Command Prompt window displays the so called prompt like C:\> or C:\Users\UserName> or in my case K:\>

Suppose you see C:\>. Type at this prompt

```
D:
```

and press ‘Enter’. The result will be the prompt D:\> (only if you have a D: partition; otherwise skip this).

Suppose the prompt is C:\Users\UserName>. Type at the prompt

```
CD ..
```

(`'CD'` space two dots) and press ‘Enter’. The prompt will be changed into C:\Users>. Do this again and the prompt will be C:\>

Again, in this text I’ll use drive D:

And for readability reasons, I use uppercase commands, e.g. `'CD'` instead of `'cd'`. Lowercase commands will do the same.

3.2 Moving into a (sub)directory

To move into a directory, use the `'CD'` command. So to move into the example directory ‘test’, type at the prompt D:\>

```
CD test
```

and press ‘Enter’. The prompt D:\test> will be displayed.

To move into the subdirectory ‘my Doc’, type at the prompt D:\test>

```
CD my Doc
```

and press ‘Enter’. You’ll see the prompt D:\test\my Doc>

To return to the directory ‘test’ again, type at the prompt D:\test\my Doc> the command

```
CD ..
```

and press ‘Enter’. The prompt D:\test> appears again.

This will do for now. Next I’ll show you techniques to batch modify ‘names’ after having explained something about pattern-matching.

4. Pattern-matching

The Windows Command Interpreter or Windows shell has a pattern-matching feature that makes operating on large numbers of ‘names’ easy. Notice that ‘names’ refer to pathnames, filenames, extensions, directories etc. The wildcard patterns (also called Glob patterns or globby patterns) are defined by two characters: the character `*` (asterisk, star) and the character `?` (question mark). Both have a special meaning to the Windows shell.

`*` matches zero or more of any characters
`?` matches any single character

When these characters are found, the shell will try to match these characters against the ‘names’, e.g. filenames in a directory.

A few examples:

`????.txt` will accept all files with the extension `'.txt'` that have filenames of four characters long such as

1234.txt
abcd.txt
A-01.txt

So to match pic001.jpg, pic002.jpg from our example directory, you could use `pic00?.jpg` or `pic???.jpg`

To get all the filenames with an undefined name length, you have to use the wildcard `*`: it will accept all filenames regardless how many characters they have.

`*.*` refers to all files.
`*.txt` refers to all files that have the extension `'.txt'`.
`*abc*.txt` refers to all files with the extension `'.txt'` that have the string `'abc'` in the filename.

To match the two example audio files musicBach.mp3 and musicBrahms.mp3, you could use (among others) `music*.mp3` or `m*.mp3` or `*.mp3`

5. Command ‘DIR’ and Glob patterns

Glob patterns can be used as parameters of a command.

Example:

- the `'DIR'` command lists all files in a directory.
- the `'DIR'` command with the parameter `*.txt` shows only files with the extension `'.txt'`

Type at the prompt (in my case: D:\test>)

```
DIR
```

and press ‘Enter’. The output contains -among other things- all the files in the current directory, the entry of the subdirectory ‘my Doc’ inclusive.

```
22-01-2016  11:06    <DIR>          .
22-01-2016  11:06    <DIR>          ..
22-01-2016  11:06    <DIR>          my Doc
22-01-2016  11:07                 1 file1.txt
22-01-2016  11:07                 1 file2.dat
22-01-2016  11:07                 1 file2.txt
22-01-2016  11:07                 1 file3.txt
23-01-2016  12:18                85 pic001.jpg
23-01-2016  12:18               123 pic002.jpg
25-01-2016  19:34             5.852 musicBach.mp3
25-01-2016  19:40             7.966 musicBrahms.mp3
```

Type at the prompt

```
DIR *.txt
```

and press ‘Enter’. The result is a list of all files with the extension `'.txt'`.

We investigate more Glob patterns and possibilities with the command `'COPY'`.

By the way, notice that the `'DIR'` command has useful switches, e.g. `/OS` (Order by Size), `/OE` (Order by Extension) and `/OD` (Order by Date). So, the command

```
DIR /OD
```

-after pressing ‘Enter’- lists all files by date. To learn more about the `'DIR'` command, type at the prompt

```
DIR /?
```

and press ‘Enter’.

Reading notes for this ebook

1. The instruction ‘Press Enter’ will be shortened to `[ENTER]`.
2. The instructions ‘type at the prompt’ will be shortened to ‘type’ and each command will start with the prompt: D:\test>

6. The ‘COPY’ command and Glob patterns

Before giving examples of the `'COPY'` command, some words on TAB completion. Type:

```
D:\test>CD my Doc [ENTER]
```

This command -as we have seen- results in the prompt

D:\test\my Doc>

To avoid much typing, use TAB completion. Type:

```
D:\test\my Doc>CD .. [ENTER]
```

The prompt will be D:\test> again. Type now:

```
D:\test>CD my
```

and press TAB. The result is:

```
D:\test>CD my Doc
```

Press Enter and you’ll see the prompt

D:\test\my Doc>

If your command does not work, try to surround names with space(s) by double quotation marks (double apostrophes).

So if this command

```
D:\test>CD my Doc [ENTER]
```

does not work, try

```
D:\test>CD "my Doc" [ENTER]
```

To be sure, I’ll use in this ebook the double quotation marks in case of names with space(s).

6.1 Copy files from the current directory into a subdirectory

To copy ‘file1.txt’ from our example directory into the subdirectory ‘my Doc’, we could type

```
D:\test>COPY file1.txt "my Doc" [ENTER]
```

To copy the files ‘file1.txt’, ‘file2.dat’, ‘file2.txt’ and ‘file3.txt’ to ‘my Doc’, use Glob patterns:

```
D:\test>COPY file?.* "my Doc" [ENTER]
```

where the question mark `?` refers to the number 1, 2 and 3 and the pattern `'.*'` matches all extensions.

6.2 Copy files from the current directory into a subdirectory in binary mode

The default copy behavior treats files as ASCII-files, as lines of text (with end-of-line characters, end-of-files etc.). The command

```
D:\test>COPY *.jpg "my Doc" [ENTER]
```

will copy the image files with extension ‘.jpg’ into the subdirectory ‘my Doc’. However, loading those images into an image viewer will probably give errors. To avoid this, you should use the binary mode: in this case the files are copied byte for byte. Simply add the switch `'/B'` to the `'COPY'` command:

```
D:\test>COPY /B *.jpg "my Doc" [ENTER]
```

6.3 Combine ASCII-files and put the result into a subdirectory

Study the following command:

```
D:\test>COPY /A file?.* "my Doc"\combinedText.txt [ENTER]
```

The text files ‘file1.txt’, ‘file2.txt’, ‘file2.dat’ and ‘file3.txt’ are concatenated. The switch `'/A'` refers to the ASCII-mode, with `'/B'` you switch to the binary mode.

6.4 Combine binary files and put the result into a subdirectory

To combine (‘concatenate’) two MP3 files (binary files!) from the current directory and copy the result to “my Doc”, type:

```
D:\test>COPY /B *.mp3 "my Doc"\CombinedBin.mp3 [ENTER]
```

The CombinedBin.mp3 contains now both mp3 files from the current directory.

6.5 Are the files copied correctly?

Add the switch `'/V'` to the `'COPY'` command to verify that the new files were written correctly.

6.6 Copy a selection of files with the ‘FOR’ loop

To copy a set of files that cannot be matched with Glob patterns easily, use the `'FOR'` command.

The `'FOR'` syntax is quite easy:

```
FOR %variable IN (set) DO command [command parameters]
```

Study the next loop:

```
D:\test>FOR %i IN (file1.txt file2.dat) DO COPY /A /V %i "my Doc" [ENTER]
```

(set) is replaced by a list of two files, that are separated by a space: (file1.txt file2.dat). The variable %i takes these list values: first %i has the value ‘file1.txt’ and then ‘file2.dat’.

If `'ROBOCOPY'` is available to you (check by typing `'ROBOCOPY /?'` at the prompt and pressing Enter), then you could avoid the `'FOR'` loop:

```
D:\test>ROBOCOPY D:\test "D:\test\my Doc" file1.txt file2.dat [ENTER]
```

To copy all text files from the current directory and all subdirectories, use the switch `'/R'` and `'*.txt'`.

```
D:\test>FOR /R %i IN (*.txt) DO COPY /A /V %i D:\tmp [ENTER]
```

(you have to make the target directory ‘tmp’ first!)

With `'ROBOCOPY'`:

```
D:\test>ROBOCOPY /S D:\test D:\tmp *.txt [ENTER]
```

With the switch `'/S'`, the `'.txt'` files from subdirectories will be copied also.

If D:\tmp does not exist, `'ROBOCOPY'` will create the directory!

`'ROBOCOPY'` has many possibilities. To give a last example:

```
D:\test>ROBOCOPY /S /E D:\test D:\tmp [ENTER]
```

copies all files and subdirectories (`'/S'`), empty ones inclusive (`'/E'`)!

7. The ‘DEL’ command and Glob patterns

Use the `'DEL'` command to delete data. Warning: handle the `'DEL'` command with care. And you should use Glob patterns cautiously with the `'DEL'` command. In special cases, the shell will ask you to confirm a deletion. But do not rely on this!

Tip: sometimes you could use the switch `'/P'` that prompts you for confirmation before deleting a file. Of course, this is not what you want when deleting many files.

7.1 Delete files from the current directory

To delete or erase file1.txt, file2.txt, file2.dat and file3.txt from our example directory D:\test, we could type:

```
D:\test>DEL file?.* [ENTER]
```

(as we saw earlier)

Another example: the command

```
D:\test>DEL *.jpg [ENTER]
```

will delete all image files with extension `'.jpg'` from our example directory.

To delete also files with the attribute read-only, you have to add the switch `'/F'` as a parameter to the `'DEL'` command.

```
D:\test>DEL /F *.jpg [ENTER]
```

7.2 Delete files from the subdirectory ‘my Doc’ -1

The command

```
D:\test>CD "my Doc" [ENTER]
```

results in the prompt

```
D:\test\my Doc>
```

To erase all files in the subdirectory ‘my Doc’, type

```
D:\test\my Doc>DEL * [ENTER]
```

The character `*` refers to all files in the subdirectory ‘my Doc’.

7.3 Delete files from the subdirectory ‘my Doc’ -2

The steps in the last example `'CD "my Doc"'` and `'DEL *'` could be combined into one line of code:

```
D:\test>CD "my Doc" & DEL * [ENTER]
```

or shorter

```
D:\test>CD m* & DEL * [ENTER]
```

where `'m*'` refers to the subdirectory ‘my Doc’.

Notice that this code does not work if another subdirectory with the first letter ‘m’ exists, e.g. myBackup.

7.3.1 An alternative

The last examples consisted of several steps, that are combined. In case of `'DEL'`, there is an easier alternative:

```
D:\test>DEL "my Doc"\* [ENTER]
```

or still shorter

```
D:\test>DEL "my Doc" [ENTER]
```

Notice that the subdirectory will not be erased; only the files will be deleted.

```
D:\test>RD "my Doc" /S [ENTER]
```

will erase the subdirectory ‘my Doc’ and all its files (`'RD'` is an abbreviation of Remove Directory; for also deleting subdirectories, the switch `'/S'` is necessary).

7.3.2 ROBOCOPY

'ROBOCOPY' is very efficient in deleting files and (sub)directories.

With the switches '/MOV /S /E' you copy the files from the source to the destination and after copying you delete the source files.

```
D:\test>ROBOCOPY /MOV /S /E D:\test D:\test_backup [ENTER]
```

With the /MOVE switch, you can delete both files and directories. Be careful with this command!

Before studying the 'REN' command in depth (chapter 9), which will show you the power of the command line, we first have a closer look at the combining of commands.

8. Passing multiple commands

In 7.3 we used the character `'&'` to execute two commands in a single line.

```
D:\test>CD "my Doc" & DEL * [ENTER]
```

The shell runs the first command (`'CD'`) and then -unconditionally- the second command (`'DEL'`). ‘Unconditionally’ means regardless the result of the `'CD'` command. So, this can be tricky. Better is to use two other passing methods with the conditional processing symbols ‘&&’ and ‘||’.

```
D:\test>CD "my Doc" && DEL * [ENTER]
```

Instead of `'&'` we use `'&&'`. It means that the shell runs the first command (`'CD'`). And only if the first command completed successfully, it runs the second command (`'DEL'`). So the following code will not run:

```
D:\test>CD "my DirNonExist" && DEL * [ENTER]
```

The third passing method uses ‘||’.

```
D:\test>CD "my DirNonExist" || DEL * [ENTER]
```

The shell will run the second command only if the first command fails. In this example it means that all files of D:\test will be deleted! So handle with care!

9. The ‘REN’ or ‘RENAME’ command

To start, the `'REN'` command has a bug: in special cases files get renamed twice (or even three times). We avoid this by a trick (see chapter 10).

9.1 Change subdirectory name

Perform this command:

```
D:\test>REN "my Doc" "my Doc Backup" [ENTER]
```

Type

```
D:\test>DIR [ENTER]
```

and you'll see that the subdirectory 'my Doc Backup' exists and 'my Doc' doesn't.

Remember that it is a good idea to surround 'names' with space(s) by double quotation marks (double apostrophes).

An alternative command is:

```
D:\test>MOVE "my Doc" "my Doc Backup" [ENTER]
```

9.2 Change file extensions

A simple example: change the extension `'.txt'` into `'.dat'`

```
D:\test>REN file3.txt file3.dat [ENTER]
```

In case of a bulk rename, you should use

```
D:\test>REN *.txt *.dat [ENTER]
```

It means that all files with the extension `'.txt'` will get the extension `'.dat'`.

To change the extension of every (!) file, use

```
D:\test>REN * *.txt [ENTER]
```

9.3 Modify filenames from the current directory: basic examples

To change the filename ‘file1.txt’ into ‘file1_1.txt’, type:

```
D:\test>REN file1.txt file1_1.txt [ENTER]
```

The `'.txt'` files in our example directory have the filenames: ‘file1.txt’, ‘file2.txt’ and ‘file3.txt’. To uppercase the first letter ‘f’ of these filenames:

```
D:\test>REN f*.txt F*.txt [ENTER]
```

Swap the parameters to lowercase the first character of a filename.

Use the wildcard `?` to change the first letter of filenames with the extension `'.txt'`.

```
D:\test>REN ?*.txt T*.txt [ENTER]
```

So ‘file1.txt’ will be renamed into ‘Tile1.txt’.

Use the wildcard ?? for the change of the first two letters of filenames with the extension `'.txt'`:

```
D:\test>REN ??*.txt Ta*.txt [ENTER]
```

So ‘file1.txt’ will be renamed into ‘Tale1.txt’.

Change the second letter of filenames with the extension `'.txt'`:

```
D:\test>REN ?i*.txt ?a*.txt [ENTER]
```

So ‘file1.txt’ will be renamed into ‘fale1.txt’.

Notice that

```
D:\test>REN *i*.txt *a*.txt
```

with an `*` at the beginning of the pattern will not work. To replace the letter ‘i’ by an ‘a’ regardless its position, we have to use the `'FOR'` loop (Chapter 10).

9.4 Truncate a filename by using ‘?’

The command

```
D:\test>REN ???*.txt ???.txt [ENTER]
```

truncates the filename of all `'.txt'` files to the first three characters (three question marks `???`). So: ‘1_file.txt’ will be ‘1_f.txt’.

9.5 Modify filenames in the subdirectory ‘my Doc’: basic example

To rename ‘doc 1.rtf’ to ‘doc_1.rtf’ in subdirectory ‘my Doc’:

```
D:\test>CD "my Doc" [ENTER]
D:\test\my Doc>REN "doc 1.rtf" doc_1.rtf [ENTER]
```

Of course, this can be done in one command as well:

```
D:\test>CD "my Doc" & REN "doc 1.rtf" doc_1.rtf [ENTER]
```

or

```
D:\test>REN "my Doc"\"doc 1.rtf" doc_1.rtf [ENTER]
```

10. More complex replacements

To replace a character regardless of its position, to add a prefix or to add a suffix, we could use the `'FOR'` loop.

It’s recommended to add the command `'LFNFOR On'` to enable Long file name support:

```
LFNFOR On & FOR %i IN (set) DO command [command parameters]
```

For brevity, we omit `'LFNFOR On'` in the next examples.

10.1 Add a prefix to filenames with the same characters at the beginning

```
D:\test>FOR %i IN (file*.txt) DO REN %i prefix_%i [ENTER]
```

The result is that e.g. ‘file1.txt’ is renamed to ‘prefix_file1.txt’.

The command

```
D:\test>FOR %i IN (file*.*) DO REN %i prefix_%i [ENTER]
```

also renames ‘file2.dat’ to ‘prefix_file2.dat’.

10.2 Add a prefix to filenames with the same extensions

The command

```
D:\test>FOR %i IN (*.txt) DO REN %i prefix_%i [ENTER]
```

makes the `'REN'` bug visible. In case of our three example files with filenames ‘file1.txt’, ‘file2.txt’ and ‘file3.txt’, the result will be:

prefix_file2.txt
prefix_file3.txt
prefix_prefix_file1.txt

To avoid the last incorrect filename, we use a little trick:

```
D:\test>FOR %i IN (*.txt) DO REN %i prefix_%i.tmp & REN *.txt.tmp *.
[ENTER]
```

What does it do? ‘file1.txt’ will be renamed to ‘prefix_file1.txt.tmp’, ‘file2.txt’ to ‘prefix_file2.txt.tmp’ etc. The command `'REN *.txt.tmp *.'` renames all extensions `'.txt.tmp'` back to `'.txt'`.

In chapter 11 we’ll use the alternative command `'FORFILES'`.

10.3 Add a suffix to filenames with the same extensions

Here we can avoid a `'FOR'` loop:

```
D:\test>REN *.txt ?????_suffix.txt.tmp & REN *.txt.tmp *. [ENTER]
```

The number of `?` equals the maximum number of characters of the original filename. So, ‘file1.txt’ will be renamed into ‘file1_suffix.txt’.

In case of renaming file1.txt, file222.txt and file33333.txt, the number of `?` characters equals the number of characters of the longest filename: file33333, i.e. 9 characters.

```
D:\test>REN *.txt ?????????_suffix.txt.tmp & REN *.txt.tmp *. [ENTER]
```

So, ‘file1.txt’ will be renamed into ‘file1_suffix.txt’, ‘file222.txt’ into ‘file222_suffix.txt’ and ‘file33333.txt’ into ‘file33333_suffix.txt’.

In chapter 11 we’ll use the alternative command `'FORFILES'`.

10.4 Substitute a character in a specific position

To substitute a character in the 1st and 3rd positions prior to the extension `'.txt'`, type:

```
D:\test>REN *.txt F?L*.txt [ENTER]
```

So ‘file1.txt’ will be renamed in ‘FiLe1.txt’ and ‘temp.txt’ into ‘FeLmp.txt’.

A 2nd or 3rd character will be added if these characters don’t exist: the file ‘1.txt’ will be changed into ‘FLL.txt’.

11. The command ‘FORFILES’

A very useful command for batch processing is `'FORFILES'`. It executes a command on each file that is selected. An interesting feature is that you are able to select files on a timestamp.

The complete syntax is:

```
FORFILES [/P Path] [/M SearchMask] [/S] [/C Command] [/D [+ | -] {date | dd}]
```

which is really puzzling. So let’s extract what we need.

```
FORFILES [/P Path] [/M SearchMask] [/C Command]
```

where `'[/P Path]'` refers to the Path to search. If we use the default, i.e. the current directory, the syntax will be still easier to understand:

```
FORFILES [/M SearchMask] [/C Command]
```

With `'[/M SearchMask]'`, you specify a search mask for selecting files. The default is `*.*` and refers to all files.

The command parameter `'[/C command]'` is used to execute some command on each file. The default is `"cmd /c echo @file"` (with double quotes!) where `'@file'` is the name of the file. Several command variables can be used in the command string. The most important commands for now are:

`'@file'`: the name of the file
`'@fname'`: the filename without extension
`'@ext'`: only the extension of the file

Let’s have a look at some examples.

11.1 Add a prefix to filenames

Type:

```
D:\test>FORFILES /M *.txt /C "cmd /c REN @file PREFIX_@file" [ENTER]
```

All text files will now have a filename that starts with “PREFIX_”.

11.2 Add a suffix to filenames

Type:

```
D:\test>FORFILES /M *.txt /C "cmd /c REN @file @fname_SUFFIX@ext" [ENTER]
```

All text files will have now a filename that ends with “_SUFFIX”.

11.3 Modifying filenames in the current directory and its subdirectories

Do you also want to change the filenames in subdirectories? Just add the `'/S'` parameter to the command.

So to extend filenames of text files in the current directory and its subdirectories with a suffix, type:

```
D:\test>FORFILES /S /M *.txt /C "cmd /c rename @file @fname_SUFFIX@ext"
[ENTER]
```

11.4 Select files on a date

Adding `'/D -dd'` to the `'FORFILES'` command makes it possible to select files older than `'dd'` days (basis: the current date). A valid `'dd'` number of days can be any number in the range of 0 to 32768(=89 years).

Make first a subdirectory ‘backup’:

```
D:\test>MD backup [ENTER]
```

To backup all text files in the current directory that were created/modified more than three days ago, type:

```
D:\test>FORFILES /M *.txt /C "cmd /c COPY @file D:\test\backup" /D -3
[ENTER]
```

Or before a specific date:

```
D:\test>FORFILES /M *.txt /C "cmd /c COPY @file D:\test\backup" /D
-01/01/2016 [ENTER]
```

(Maybe you have to adapt the date format, e.g 01-01-2016)

To backup all text files in the current directory created/modified after 1st January 2016, type

```
D:\test>FORFILES /M *.txt /C "cmd /c COPY @file D:\test\backup" /D
+01/01/2016 [ENTER]
```

(with the plus sign before the date. The ‘+’ could be omitted because it is default).

11.5 List files and export the list to a file

The switch `'/D'` can be put anywhere in the command, that can be fine for readability. Example: to select all files in the current directory that were created/modified more than a day ago and export the result to a file, type:

```
D:\test>FORFILES /M *.* /D -1 /C "cmd /C ECHO @file" > D:\test\result.txt
[ENTER]
```

12. Batch files

If a command line is too long, it will be difficult to read and to understand. In this case, you should use a batch file. Some examples.

12.1 Add a sequential number to filenames: prefix

Open Notepad, enter the following code and save the file to ‘number1.bat’ (in directory ‘test’)

```
@echo off
setlocal EnableDelayedExpansion
SET i=0
FOR /f "usebackq tokens=" %%f in (*.txt) DO (
    SET /a i+=1
    REN "%%f" "!i!%%f.tmp"
)
REN *.tmp *.
```

To run this file, type at the prompt the filename ‘number1.bat’ and press Enter.

```
D:>\test\number1.bat [ENTER]
```

Some explanations:

a. The command `@echo off` means: ‘do not display the commands in this file on the screen’
b. `setlocal EnableDelayedExpansion`: the value of variables that are modified inside the `'FOR'` command are taken by enclosing their names in exclamation-marks. Otherwise, variable i will always has the value 0 (in this case: ‘SET i=0’).
c. For filenames with spaces, add `"usebackq tokens="`.
d. A variable in a batch file is represented by `%%variable` (instead of `%variable`). So with two percent signs instead of one.

12.2 Add a sequential number to filenames: suffix

Open Notepad, enter the following code and save the file to ‘number2.bat’ (in directory ‘test’)

```
@echo off
setlocal EnableDelayedExpansion
SET i=0
FOR /f "usebackq tokens=" %%f in (*.txt) DO (
    SET /a i+=1
    REN "%%f" "%%~nf!i!.tmp"
)
REN *.tmp *.txt
```

To run this file, type at the prompt the filename ‘number2.bat’ and press Enter.

```
D:>\test\number2.bat [ENTER]
```

Explanation:

`%%~nf` expands `%%f` to a filename only. For more info, enter `FOR /?` at the prompt.
For filenames with spaces, add `"usebackq tokens="`.

12.3 Add a sequential number as suffix to filenames in sorted order

Open Notepad, enter the following code and save the file to ‘number3.bat’ (in directory ‘test’)

```
@echo off
setlocal EnableDelayedExpansion
SET i=0
FOR /f "usebackq tokens=" %%f in (DIR "*.txt" /B /O:N) DO (
    SET /a i+=1
    REN "%%f" "%%~nf!i!.tmp"
)
REN *.tmp *.txt
```

To run this file, type at the prompt the filename ‘number3.bat’ and press Enter.

```
D:>\test\number3.bat [ENTER]
```

Explanation:
the command `'DIR /B'` results in a sorted bare list of files, sorted by the switch `'/O:N'`.

12.4 Command line parameters batch files

Batch files are handy tools. However, they can have security issues, e.g. when using command line arguments. I recommend to read :

www.robvanderwoude.com/battech_inputvalidation_commandline.php

The website www.robvanderwoude.com gives a lot of information, including a tutorial on batch files.

13. Tips and tricks

Last but not least: in the next chapters, I’ll give you some useful tips and tricks.

13.1 Make files read-only

To set the file attribute to read-only (as the name suggests: the file can be read from, but not written to), use the command `'ATTRIB'`:

```
D:\test>ATTRIB +r "my Doc"\doc2.rtf [ENTER]
```

Notice the parameter `'+r'` (plus sign). Of course, with Glob patterns you can easily change more than one file. In the next example all files with the extension `'.rtf'` in subdirectory ‘my Doc’ are made read-only:

```
D:\test>ATTRIB +r "my Doc"\*.rtf [ENTER]
```

13.2 Remove read-only attribute

To remove the read-only attribute of files, use the command `'ATTRIB'` with parameter ‘`-r`‘ (minus sign) :

```
D:\test>ATTRIB -r "my Doc"\*.rtf [ENTER]
```

13.3 Show read-only files

Use the `'DIR'` command with the switch `'/A:R'` to show read-only files:

```
D:\test>DIR /A:R [ENTER]
```

13.4 (un)Hiding a file

To hide a file in the subdirectory ‘my Doc’ with the command `'ATTRIB'` and parameter ‘+h’ (plus sign):

```
D:\test>ATTRIB +h "my Doc"\file1.txt [ENTER]
```

To unhide a file:

```
D:\test>ATTRIB -h "my Doc"\file1.txt [ENTER]
```

So `'ATTRIB'` and parameter ‘-h’ (minus sign)

Delete all hidden files (Is this really a clever action? Never use this command unless you are sure what you do!):

```
DEL /A:H "my Doc"\*.txt [ENTER]
```

You can also hide a directory with the `'ATTRIB'` command!

13.5 Secure file deletion

In Linux, I use the ‘shred’ command to delete files permanently. Via the Windows Command Line you can also do some secure file deletion: firstly you turn an existing file into a zero-byte file, secondly you delete the file via the `'DEL'` command and thirdly you run the `'CIPHER'` command:

```
D:\test>TYPE nul >file1.txt && DEL file1.txt [ENTER]
```

(Use the `'FOR'` command to delete more than one file).

After deleting your files, you run the `'CIPHER'` command that will overwrite deleted files, i.e. free disk space, maybe freeing up some extra disk space.

```
D:\test>CIPHER /w:D:\test [ENTER]
```

In one command line:

```
D:\test>TYPE nul >file1.txt && DEL file1.txt && CIPHER /w:D:\test [ENTER]
```

An alternative is the command line tool SDelete:

https://technet.microsoft.com/en-us/sysinternals/bb897443

Tip: run

```
D:\test>CIPHER /? [ENTER]
```

for lots of information on the `'CIPHER'` command!

13.6 Copy from the Windows Command Prompt to the Clipboard

If you enable QuickEdit Mode, you can copy text from the Windows Command Prompt and paste it in a document e.g. Notepad. To enable Quickedit Mode,

a. do a right-mouse click on the title bar: a dialog pops up
b. select Properties
c. check the ‘QuickEdit Mode’ box
d. close the dialog

Select some text by dragging a box around your text and do a right-mouse click (or press Enter): the text is copied to the Clipboard. With CTRL+V you can paste the text into e.g. Notepad.

13.7 Copy output command to the Clipboard

To copy the output of a command to the Clipboard, type:

```
D:\test>DIR | CLIP [ENTER]
```

(notice that this will not work on XP)

13.8 Save output command to a file

For your convenience, here the command again to save the output of a command into a file using the redirection operator >

```
D:\test>DIR > result.txt [ENTER]
```

13.9 ‘TASKLIST’ and ‘TASKKILL’

To end one or more processes by the command `'TASKKILL'`, you need a process id (PID) or a so called ‘imagename’. To get this information, you have to run `'TASKLIST'`.
Example: start notepad and type:

```
D:\test>TASKLIST [ENTER]
```

In my case, notepad.exe (notice the column name: ‘imagename’) has process id 4324. You can end this process in two ways. The first way is with the switch `/IM` and ‘imagename’ as command parameter:

```
D:\test>TASKKILL /IM notepad.exe [ENTER]
```

The second way has the switch `/PID` and the process id as parameter:

```
D:\test>TASKKILL /PID 4324 [ENTER]
```

In both cases, multiple processes can be killed. In case of `/IM` you can use Glob patterns.

The command

```
D:\test>TASKLIST | FIND "cmd" [ENTER]
```

displays the PID of your CMD-window in the second column.

Notice that home editions of Windows do not have `'TASKKILL'`. You should use `'TSKILL'` instead.

13.10 Use functions keys

The most popular function keys in the Windows Command Line window for me are:

`F3`: it pastes the last command (see also the explanation of `F8`)
`F7`: it shows a list of previous commands: select a command and press Enter in order to run the selected command.
`F8`: press `F8` one or more times for pasting commands from your command history; this can also be done by up and down arrow keys.

14. Command Prompt replacement

ConEmu is an interesting alternative for the Windows' built-in Command Prompt. It had a lot of options and is highly customizable (do a right-mouse click on the title bar of ConEmu).

Download the .7z file from

https://conemu.github.io/

unpack it into any directory and double-click the ConEmu.exe or ConEmu64.exe with the left mouse button.

15. Help

All commands are documented on the web (2) but also offline. Type at the prompt a command with the switch `/?`

If you type

```
D:\test>CD /? [ENTER]
```

you’ll get information on the command `'CD'`.

To find which commands are available, type:

```
D:\test>HELP [ENTER]
```

16. Why use the Windows Command Line?

Is it necessary to use Windows Command Line or batch scripts? It depends. There are several tasks that can be done better with the Windows Command Line than within Windows Explorer.

However, if your tasks are rather complex, I would recommend learning a programming language like C or newLisp.

Or if you like a very user-friendly programming software, I suggest the very nice RAD-tool with a relatively easy scripting language: Neobook (www.neosoftware.com). I use Neobook for many years now, when making Windows applications (e.g. my MC Musiceditor - www.mcmusiceditor.com). Many plugins that enhances the Neobook applications, are available as well. Of course, within Neobook you can run Windows commands and batch files as we have seen in this ebook.

So give Neobook a try, if you will develop your own Windows tools (3).

17. Colophon

The text of this ebook was put in separate HTML-files, that were converted into the MOBI-format with the KindleGen command line tool of Amazon.

If you enjoyed this ebook, please write a review on Amazon. If you would like to comment, send an email to

`reinier[dot]maliepaard[at]gmail[dot]com`

Thanks for buying this ebook!

18. Notes

(1) Read my former ebook ‘Switching lanes: heading towards Linux (Knoppix)’, available on amazon.com
(2) See technet.microsoft.com/en-us/library/bb490890.aspx and also ss64.com/nt for the distinction between internal and external commands.
(3) Neobook applications run fine with Wine under Linux.