

(0) the first and last lines of a batch script

```
@echo off
setlocal
setlocal enableextensions
setlocal enabledelayedexpansion

...

exit /b
```

(1) select characters between the M-th and the N-th character

```
set M=2
set N=3
set A=0123456789      set B=%A:~2,-3%      "23456"
                      set B=!A:~%M%,-%N%!  "23456"
```

(2) select the last N characters

```
set N=3
set A=0123456789      set B=%A:~-3%      "89"
                      set B=!A:~-%N%!      "89"
```

(3) replace

```
A=123123123          set B=%K:3=X%      "12X12X12X"
```

(4) cut out characters between the M-th and the N-th character

```
set M=2
set N=3
set A=0123456789      set B=!A:~0,%M%!!A:~-%N%!  "12789"
```

(5) cut out N characters from position M

```
set M=2
set N=3
set A=0123456789          set K=!A:~%M%,%N%!
                          set L=!A:%K%=!          "012789"
```

(6) delayed variable evaluation

```
set /a N=0 >NUL && for /L %i in (1,1,3) do \
    @( set /a N=N+1 >NUL && echo %i %N% )

1 0
2 0
3 0

set /a N=0 >NUL && for /L %i in (1,1,3) do \
    @( set /a N=N+1 >NUL && echo %i !N! )

1 1
2 2
3 3
```

(7) oneliners and batch scripts: %i vs. %i

oneliner

```
set /a N=0 >NUL && for /L %i in (1,1,3) do \
    @( set /a N=N+1 >NUL && echo %i !N! )
```

batch script

```
set /a N=0 >NUL && for /L %%i in (1,1,3) do \
    @( set /a N=N+1 >NUL && echo %%i !N! )

set /a N=0 >NUL
for /L %i in (1,1,3) do (
    set /a N=N+1 >NUL
    echo %i !N!
)
```

(8) set numerical variables

```
set /a N=1
set /a N=N+1    "2"
```

(8.1) arithmetic operators

()	- grouping
! ~ -	- unary operators
* / %	- arithmetic operators
+ -	- arithmetic operators
<< >>	- logical shift
&	- bitwise and
^	- bitwise exclusive or
	- bitwise or
= *= /= %= += -=	- assignment
&= ^=  = <<= >>=	

arithmetic operators/expressions have to be quoted

```
set /a A="8^4"    or    set /a A=8^^4
```

(9) prompt for input

```
set /p a=input:
```

(10) numerical for-loop

```
for /L %i in (0,2,4) do @echo %i

0
2
4
```

(11) loop over files

```
for %i in (*.*) do @echo %i
```

(12) loop over folders

```
for /D %i in (c:\windows\*) do @echo %i
```

(13) loop over %PATH% == search for a file in %PATH%

```
for /D %%i in (cmd.exe) do @echo %%~$PATH:i
```

(14) recursive loop to find files

```
for /R c:\windows %%i in (*.exe) do @echo %%i
```

(15) a very simple "grep -r"

```
@echo off
for /R c:\pfad %%i in (*.cmd) do (
  type %%i | find "PATTERN" >NUL
  if not errorlevel 1 echo %%i
)
```

(16) read a file

```
k1.txt:      1,2,A,4,5
             1,2,B,4,5
             1,2,C,4,5
```

```
for /F "usebackq delims=, tokens=1,2,3,4" %%i in (k1.txt) \
do @(echo %k)
```

```
A
B
C
```

read text files with leading white spaces

```
for /F "usebackq delims=§ tokens=*" %%i in (k1.txt) \
do @(echo %k)
```

(17) if-then-else

```
set A=1
if _%A%_==_1_ (
  echo x
) else (
  echo y
)
```

(18) touch a file

```
copy /Y NUL "file.txt" > NUL
```

(19) quoting

```
echo Usage
echo      %~nx0% ^<filename^>
echo.
```

(20) variable modifiers I in a for-loop

```
%~I      - expands %I removing any surrounding quotes (")
%~fI     - expands %I to a fully qualified path name
%~dI     - expands %I to a drive letter only
%~pI     - expands %I to a path only
%~nI     - expands %I to a file name only
%~xI     - expands %I to a file extension only
%~sI     - expanded path contains short names only
%~aI     - expands %I to file attributes of file
%~tI     - expands %I to date/time of file
%~zI     - expands %I to size of file
%~$PATH:I - searches the directories in PATH
```

(21) execute external commands and catch the output

```
@echo off
for /F "usebackq delims=§ tokens=*" %%i in ('dir') do (
    echo %%i
)
```

(22) redirect STDIN and STDERR

```
dir hurbel 1>NUL 2>&1
```

(23) obtain user or admin rights (Vista+Win7)

```
whoami /groups | find "S-1-16-12288" > nul
if errorlevel 1 goto :restricted
REM .. do admin stuff here ..
:restricted
REM .. do user stuff here ..
```

(24) pseudo functions

```
@echo off
setlocal

goto :MAIN

:FUNC1
echo FUNC1 [%ARGV1%]
goto :%FUNC1BK%

:FUNC2
echo FUNC2 [%ARGV1%]
goto :%FUNC2BK%

:MAIN

set FUNC1BK=HERE1 && set ARGV1=parm1
goto :FUNC1
:HERE1

set FUNC2BK=HERE2 && set ARGV1=parm2
goto :FUNC2
:HERE2

set FUNC1BK=HERE3 && set ARGV1=parm3
goto :FUNC1
:HERE3

set FUNC2BK=HERE4 && set ARGV1=parm4
goto :FUNC2
:HERE4
```

(25) functions

```
@echo off
goto :main

::local variables
:func1
    setlocal
    set A=1
    echo func%A% %*
goto :EOF

::global variables
:func2
    set A=2
    echo func%A% %*
goto :EOF

::returns a value
:handleargs
    set ARGV=0
    :l1
        set /A ARGV+=1
        set ARGV[%ARGV]=%1
        shift
        if not _%1_==__ goto :l1
exit /b %ARGV%

:main
setlocal

call :handleargs %*
set ARGV=%ERRORLEVEL%
for /L %%i in (1,1,%ARGV%) do (
    echo ARGV[%%i]=[!ARGV[%%i]!]
)

set A=0
echo main%A%
call :func1 p11
echo main%A%
call :func2 p21
echo main%A%

exit /b 0
```

(26) lowercase functions

```
@echo off

set M=%1
set X=ABCDEFGHIJKLMNOPQRSTUVWXYZ
set Y=abcdefghijklmnopqrstuvwxyz

goto :MAIN

:LOWERCASE1
  A=%X%
  B=%Y%
  set PA=!A:~0,1!
  set A=!A:~1!
  set PB=!B:~0,1!
  set B=!B:~1!
  set M=!M:%PA%=%PB%!
if not !_!X!_==__ goto :LOWERCASE1
goto :EOF

REM a more "simple" solution using the behaviour of Windows
REM to ignore the case when comparing characters
:LOWERCASE2
  set B=%Y%
  set A=!B:~0,1!
  set B=!B:~1!
  set T=!T:%A%=%A%!
if NOT !_!B!_==__ goto :LOWERCASE2
goto :EOF

:MAIN
call :LOWERCASE1 %M%
echo %M%
```