SET

Display, set, or remove CMD environment variables. Changes made with SET will remain only for the duration of the current CMD session.

```
Syntax
      SET variable
      SET variable=string
      SET /A "variable=expression"
      SET "variable="
      SET /P variable=[promptString]
      SET "

Key
   variable    : A new or existing environment variable name e.g. _num
   string      : A text string to assign to the variable.
   expression  : Arithmetic expression
```

Arithmetic expressions (SET /a)
The expression to be evaluated can include the following operators:

```
   +   Add                  set /a "_num=_num+5"
   +=  Add variable         set /a "_num+=5"
   -   Subtract (or unary)set /a "_num=_num-5"
   -=  Subtract variable    set /a "_num-=5"
   *   Multiply             set /a "_num=_num*5"
   *=  Multiply variable    set /a "_num*=5"
   /   Divide               set /a "_num=_num/5"
   /=  Divide variable      set /a "_num/=5"
   %   Modulus              set /a "_num=5%%2"
   !   Logical negation  0 (FALSE) ⇨ 1 (TRUE) and any non-zero value (TRUE) ⇨ 0 (FALSE)
   ~   One's complement (bitwise negation)
   &   AND                  set /a "_num=5&3"    0101 AND 0011 = 0001 (decimal 1)
   &=  AND variable         set /a "_num&=3"
   |   OR                   set /a "_num=5|3"    0101 OR 0011 = 0111 (decimal 7)
   |=  OR variable          set /a "_num|=3"
   ^   XOR                  set /a "_num=5^3"    0101 XOR 0011 = 0110 (decimal 6)
   ^=  XOR variable         set /a "_num=^3"
   <<  Left Shift.    (sign bit ⇨ 0)
   >>  Right Shift.   (Fills in the sign bit such that a negative number always remains negative.)
                       Neither ShiftRight nor ShiftLeft will detect overflow.
   <<= Left Shift variable     set /a _num<<=2
   >>= Right Shift variable    set /a _num>>=2

  ( )  Brackets group expressions   set /a "_num=(2+3)*5"
   ,   Commas separate expressions  set /a "_num=2,_result=_num*5"

See Arithmetic examples below and this forum thread for more.
also see SetX, VarSearch and VarSubstring for more on variable manipulation.
```

Variable names are not case sensitive but the contents can be.

The number one problem people run into with SET is having extra spaces around either the variable name or the *string*, SET is not forgiving of extra spaces like many other scripting languages.
Variables can contain spaces, variable *names* can also contain spaces, but this is not recommended.

Display a variable:

Type SET without parameters to display all the current environment variables.

Type SET with a variable name to display that variable
`SET _department`
or use ECHO:
`ECHO [%_department%]`

The SET command invoked with a string (and no equal sign) will display a wildcard list of all matching variables

Display variables that begin with 'P':
`SET p`

Display variables that begin with an underscore
```
SET _
```

Set a variable:

Example of storing a text string:

```
C:\> SET _dept=Sales and Marketing
C:\> set _
_dept=Sales and Marketing
```

One variable can be based on another, but this is not dynamic
E.g.

```
C:\> set xx=fish
C:\> set msg=%xx% chips
C:\> set msg
msg=fish chips

C:\> set xx=sausage
C:\> set msg
msg=fish chips

C:\> set msg=%xx% chips
C:\> set msg
msg=sausage chips
```

Avoid starting variable names with a number, this will avoid the variable being mis-interpreted as a parameter
```
%123_myvar% < > %1 23_myvar
```

To display undocumented system variables:

```
   SET "
```

Prompt for user input

The /P switch allows you to set a variable equal to a line of input entered by the user.
The Prompt string is displayed before the user input is read.

```
@echo off
Set /P _dept=Please enter Department || Set _dept=NothingChosen
If "%_dept%"=="NothingChosen" goto :sub_error
If /i "%_dept%"=="finance" goto sub_finance
If /i "%_dept%"=="hr" goto sub_hr
goto:eof

:sub_finance
echo You chose the finance dept
goto:eof

:sub_hr
echo You chose the hr dept

:sub_error
echo Nothing was chosen
```

The Prompt string can be empty. If the user does not enter anything (just presses return) then the variable will be unchanged and an errorlevel will be set.

To place the first line of a file into a variable:

```
Set /P _MyVar=<MyFilename.txt
```

The CHOICE command is an alternative to SET /P

Variable names with spaces

A variable can contain spaces and also the variable name itself may contain spaces, therefore the following assignment:
`SET _var =MyText`
will create a variable called `"_var "` - note the trailing space

To avoid problems with extra spaces, issue SET statements in parentheses, like this:

```
(SET _department=Some Text)
```
Alternatively:
```
SET "_department=Some Text"
```

Note: To actually include a bracket in the variable, use an escape character.

The SET command will set ERRORLEVEL to 1 if the variable name is not found in the current environment.
This can be detected using the IF ERRORLEVEL command

Delete a variable

Type SET with just the variable name and an equals sign:

```
SET _department=
```

Better still, to be sure there is no trailing space after the = use:
```
(SET _department=)
  or
SET "_department="
```

Arithmetic expressions (SET /a)

Placing expressions in "quotes" is optional for simple arithmetic but required for any expression using logical operators.

Any SET /A calculation that returns a fractional result will be rounded down to the nearest whole integer.

Examples:

```
   SET /A "_result=2+4"
   (=6)

   SET /A "_result=5"
   (=5)
   SET /A "_result+=5"
   (=10)

   SET /A "_result=2<<3"
   (=16)   { 2 Lsh 3 = binary 10 Lsh 3 = binary 10000 = decimal 16 }

   SET /A "_result=5%%2"
   (=1)    { 5/2 = 2 + 2 remainder 1 = 1 }
```

In a batch script, the Modulus operator (`%`) must be doubled up to (`%%`).

SET /A will treat any character string in the expression as an environment variable name. This allows you to do arithmetic with environment variables without having to type any % signs to get the values. `SET /A _result=5 + _MyVar`

Leading Zero will specify Octal

Numeric values are decimal numbers, unless prefixed by
`0x` for hexadecimal numbers,
`0` for octal numbers.

So 0x12 = 022 = 18 decimal

The octal notation can be confusing - all numeric values that start with zeros are treated as octal but 08 and 09 are not valid octal digits.
This can cause errors when performing date arithmetic. For example `SET /a _day=07` will return the value 7, but `SET /a _day=09` will return an error.

Permanent changes

Changes made using the SET command are NOT permanent, they apply to the current CMD prompt only and remain only until the CMD window is closed.
To permanently change a variable at the command line use SetX
or with the GUI - Control Panel | System | Environment | System/User Variables

Changing a variable permanently with SetX will not affect any CMD prompt that is already open.
Only new CMD prompts will get the new setting.

You can of course use SetX in conjunction with SET to change both at the same time:

```
Set _Library=T:\Library\
SetX _Library T:\Library\ /m
```

Change the environment for other sessions

Neither SET nor SetX will affect other CMD sessions that are already running on the machine . This as a good thing, particularly on multi-user machines, your scripts won't have to contend with a dynamically changing environment while they are running.

It is possible to add permanent environment variables to the registry (`HKCU\Environment`), but this is an undocumented (and likely unsupported) technique and still it will not take effect until the users next login.

System environment variables can be found in the registry here:
`HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Environment`

Advanced usage - CALL SET

The CALL SET syntax allows a variable substring to be evaluated, read the CALL page for more detail on this technique:

```
 SET start=10
 SET length=9
 SET string=The quick brown fox jumps over the lazy dog
 CALL SET substring=%%string:~%start%,%length%%%
 ECHO (%substring%)
```

Autoexec.bat

Any SET statement in c:\autoexec.bat may be parsed at boot time
Variables set in this way are not available to 32 bit gui programs - they won't appear in the control panel.
They will appear at the CMD prompt.

If autoexec.bat CALLS any secondary batch files, the additional batch files will NOT be parsed at boot.
This behaviour can be useful on a dual boot PC.

SET is an internal command. If Command Extensions are disabled all SET commands are disabled other than simple assignments like:
_variable=MyText

The CMD shell will fail to read an environment variable if it contains more than 8,191 characters.

# I got my mind set on you
# I got my mind set on you... - George Harrison

Related:

Syntax - Environment Variables - List of default variables
CALL - Evaluate environment variables
SETX - Set an environment variable permanently.
SETLOCAL - Begin localisation of environment variable changes
ENDLOCAL - End localisation of environment changes, use to return values
EXIT - Set a specific ERRORLEVEL
Parameters - get a full or partial pathname from a command line variable.
PATH - Change the %PATH% environment variable.
PATHMAN - Resource Kit utility for modification of both the system and user paths. Pathman can resolve many problems and can improve performance by removing duplicate paths. For details see Pathman.wri
REG - Read or Set Registry values
REGEDIT - Import or export registry settings
WMIC ENVIRONMENT - Set environment vars through WMI
StackOverflow - Storing a Newline in a variable
Powershell: Set-Variable - Set a variable and a value (set/sv)
Powershell: Read-Host - Prompt for user input
Equivalent bash command (Linux): env - Display, set, or remove environment variables

بيع واشري كلشي في
AVITO.MA
Robe
350 DH
Voiture
140 000 DH
Machine
à laver
3 000 DH
DEPOSEZ
GRATUITEMENT
VOS ANNONCES