Parameters

A parameter (or argument) is any value passed into a batch script:

```
C:> MyScript.cmd January 1234 "Some value"
```

Parameters may also be passed to a subroutine with CALL:

```
CALL :my_sub 2468
```

You can get the value of any parameter using a % followed by it's numerical position on the command line. The first item passed is always %1 the second item is always %2 and so on

`%*` in a batch script refers to all the arguments (e.g. %1 %2 %3 %4 %5 ...%255)

Parameter Extensions

When a parameter is used to supply a filename then the following extended syntax can be applied:

we are using the variable %1 (but this works for any parameter)

`%~f1` Expand %1 to a Fully qualified path name - `C:\utils\MyFile.txt`

`%~d1` Expand %1 to a Drive letter only - `C:`

`%~p1` Expand %1 to a Path only e.g. `\utils\` this includes a trailing `\` which may be interpreted as an escape character by some commands.

`%~n1` Expand %1 to a file Name without file extension `C:\utils\MyFile` or if only a path is present (with no trailing backslash\) - the last folder in that path.

`%~x1` Expand %1 to a file eXtension only - `.txt`

`%~s1` Change the meaning of `f`, `n`, `s` and `x` to reference the Short 8.3 name (if it exists.)

`%~1` Expand %1 removing any surrounding quotes (")

`%~a1` Display the file attributes of %1

`%~t1` Display the date/time of %1

`%~z1` Display the file size of %1

`%~$PATH:1` Search the PATH environment variable and expand %1 to the fully qualified name of the first match found.

The modifiers above can be combined:

`%~dp1` Expand %1 to a drive letter and path only

`%~sp1` Expand %1 to a path shortened to 8.3 characters

`%~nx2` Expand %2 to a file name and extension only

These parameter/ argument variables are always denoted with a single leading `%`

This is unlike regular variables which have both leading and trailing `%'s` such as `%variable%`, or FOR command variables which use a single leading `%` on the command line or a double leading `%%` when used in a batch file.

When writing batch scripts it's a good idea to store the values in a variable `SET _LogFile=%~dp1`, the rest of the script can then refer to the easy-to-read variable name `%_LogFile%` This will also make life easier if you later need to change around the order of the parameters.

Passing by Reference

In addition to passing numeric or string values on the command line, it is also possible to pass a variable name and then use the variable to transfer data between scripts or subroutines. Passing by reference is a slightly more advanced technique but can be particularly useful when the string contains characters that are CMD delimiters or quotes.

Links relative to the Batch Script

You can get the pathname of the batch script itself with `%0`, parameter extensions can be applied to this so `%~dp0` will return the Drive and Path to the batch script e.g. `W:\scripts\` and `%~f0` will return the full pathname `W:\scripts\mybatch.cmd`

You can refer to other files in the same folder as the batch script by using this syntax:

```
  CALL %0\..\SecondBatch.cmd
```

This can even be used in a subroutine, `Echo %0` will give the call label but, `echo "%~nx0"` will give you the filename of the batch script.

When the %0 variable is expanded, the result is enclosed in quotation marks.

Use `%~a1` to display the Extended Attributes of a file.

FOR's `%%~aI` recognizes 9 NTFS file attributes. The expansion of a file attribute produces a series of 9 dashes, with each recognized attribute replacing a dash with a letter. A file with no recognized attributes or with none set will expand to 9 dashes like this: `---------`

```
 Attribute                    Expansion
 FILE_ATTRIBUTE_DIRECTORY     d--------
 FILE_ATTRIBUTE_READONLY      -r-------
 FILE_ATTRIBUTE_ARCHIVE       --a------
 FILE_ATTRIBUTE_HIDDEN        ---h-----
 FILE_ATTRIBUTE_SYSTEM        ----s----
 FILE_ATTRIBUTE_COMPRESSED    -----c---
 FILE_ATTRIBUTE_OFFLINE       ------o--
 FILE_ATTRIBUTE_TEMPORARY          -------t-
 FILE_ATTRIBUTE_REPARSE_POINT --------l
 FILE_ATTRIBUTE_NORMAL        ---------
```

Other NTFS attributes not recognised by `%%~aI` can be read using FSUTIL usn command:

```
FILE_ATTRIBUTE_ENCRYPTED
FILE_ATTRIBUTE_NOT_CONTENT_INDEXED
FILE_ATTRIBUTE_SPARSE_FILE
```

Example: Expansion of a file with the Hidden and System attributes:
`---hs----`

FOR parameters

The FOR command creates parameter variables which are identified with a letter rather than a number (e.g. `%%G`).

The Parameter Expansions described above can also be applied to these.
To avoid confusion between the two sets of letters you may wish to avoid using the letters (a, d, f, n, p, s, t, x, z) as FOR parameters or just choose a FOR parameter letter thats UPPER case.
So for example in a reference like `%%~fG` the `%%G` is the FOR parameter, and the `~f` is the Parameter Expansion.

Examples:

Pass parameters from one batch to another:

```
   MyBatch.cmd SMITH 100
```

Or as part of a CALL :

```
   CALL MyBatch.cmd SMITH 100
```

Passing values from one part of a script to another:

```
   :: Using CALL to jump to a subroutine
   CALL :s_staff SMITH 100

   :: Calling a subroutine from a FOR command
   FOR /F %%G IN ('DIR /b *.*') DO call :s_subroutine %%G
```

“A gift is pure when it is given from the heart to the right person at the right time and at the right place, and when we expect nothing in return” ~ The Bhagavad Gita

Related:

Bug when using `~s` to display short file/folder names
CALL - Call one batch program from another.
CMD - Start a new DOS shell (cmd.exe)
IF - Test that required inputs are in place (not NULL)
FOR - Conditionally perform a command several times.
SETLOCAL - Control the visibility of environment variables
SHIFT - Shift the position of replaceable parameters in a batch file.
StackOverflow - How does the Windows Command Interpreter (CMD.EXE) parse scripts?
Equivalent bash command (Linux): dirname - Convert a full pathname to just a path.