FINDSTR

Search for strings in files.

```
Syntax
      FINDSTR [options] [/F:file] [/C:string] [/G:file]
        [/D:DirList] [/A:color] [/OFF[LINE]] [string(s)] [pathname(s)]

      FINDSTR [options] [/F:file] [/R] [/G:file]
        [/D:DirList] [/A:color] [/OFF[LINE]] [string(s)] [pathname(s)]
Key
   string      Text to search for.
   pathname(s) The file(s) to search.
   /C:string   Use string as a literal search string.
   /R          Use string as a regular expression.
   /G:file     Get search string from a file (/ stands for console).
   /F:file     Get a list of pathname(s) from a file (/ stands for console).
   /A:color    Display filenames in colour (2 hex digits)
   /d:dirlist  Search a comma-delimited list of directories.

options may be any combination of the following switches:

   /I   Case-insensitive search.
   /S   Search subfolders.
   /P   Skip any file that contains non-printable characters
   /OFF[LINE] Do not skip files with the OffLine attribute set.
   /L   Use search string(s) literally.
   /B   Match pattern if at the Beginning of a line.
   /E   Match pattern if at the END of a line.
   /X   Print lines that match exactly.
   /V   Print only lines that do NOT contain a match.
   /N   Print the line number before each line that matches.
   /M   Print only the filename if a file contains a match.
   /O   Print character offset before each matching line.
```

If more than one file is searched, the results will be prefixed with the filename where the text was found.

Option syntax

Options can be prefixed with either `/` or `–`
Options may also be concatenated after a single / or -. However, the concatenated option list may contain at most one multicharacter option such as `OFF` or `F:`, and the multi-character option must be the last option in the list.

The following are all equivalent ways of expressing a case insensitive regex search for any line that contains both "hello" and "goodbye" in any order

```
    /i /r /c:"hello.*goodbye" /c:"goodbye.*hello"
    -i -r -c:"hello.*goodbye" /c:"goodbye.*hello"
    /irc:"hello.*goodbye" /c:"goodbye.*hello"
```

Regular Expressions (Search for patterns of text)

FINDSTR can use the following metacharacters which have special meaning either as an operator or delimiter. FINDSTR support for regular expressions is limited and non-standard, only the following metacharacters are supported:

```
 .         Wildcard: any character

 *         Repeat: zero or more occurances of previous character or class

 ^         Line position: beginning of line
 $         Line position: end of line

 [class]   Character class: any one character in set
 [^class]  Inverse class: any one character not in set
```

```
[x-y]     Range: any characters within the specified range

\x        Escape: literal use of metacharacter x

\<xyz     Word position: beginning of
xyz\>     Word position: end of word
```

Metacharacters are most powerful when they are used together. For example, the combination of the wildcard character (.) and repeat (*) character is similar in effect to the filename wildcard (*.*)

```
.*        Match any string of characters
```

The .* expression may be useful within a larger expression, for example `a.*b` will match any string beginning with A and ending with B.

FINDSTR does not support alternation with the pipe character (`|`) multiple Regular Expressions can be separated with spaces, just the same as separating multiple words (assuming you have not specified a literal search with /C) but this may not be useful if the regex itself contains spaces.

Regex Line Position anchors `^` and `$`

`^` matches beginning of input stream as well as any position immediately following a <LF>. Since FINDSTR also breaks lines after <LF>, a simple regex of "^" will always match all lines within a file, even a binary file.

`$` macthes any position immediately preceding a <CR>. This means that a regex search string containing `$` will never match any lines within a Unix style text file, nor will it match the last line of a Windows text file if it is missing the EOL marker of <CR><LF>.

Note - As detailed further below, piped and redirected input to FINDSTR may have <CR><LF> appended that is not in the source. This can impact a regex search that uses `$`.

Any search string with characters before `^` or after `$` will always fail to find a match.

Positional Options `/B /E /X`

The positional options work the same as `^` and `$`, except they also work for literal search strings.

`/B` functions the same as ^ at the start of a regex search string.

`/E` functions the same as $ at the end of a regex search string.

`/X` functions the same as having both ^ at the beginning and $ at the end of a regex search string.

FINDSTR - Escapes and Length limits - More detail of how to use search strings that include quotes and/or backslashes. Also maximum Search String length limits vary with OS version.

FINDSTR - Searching across line breaks

Regex character class ranges [x-y]

Character class ranges do not work as expected. See this Q/A on Stack Exchange: Why does findstr not handle case properly (in some circumstances)?

The problem is FINDSTR does not collate the characters by their byte code value (commonly thought of as the ASCII code, but ASCII is only defined from 0x00 - 0x7F). Most regex implementations would treat [A-Z] as all upper case English capital letters. But FINDSTR uses a collation sequence that roughly corresponds to how SORT works. So [A-Z] includes the complete English alphabet, both upper and lower case (except for "a"), as well as non-English alpha characters with diacriticals.

The FINDSTR regex sorts lower case before upper case. So `findstr /nrc:"^[A-a]"` will find nothing, but `findstr /nrc:"^[a-A]"` will match.

Default type of search: Literal vs Regular Expression

`/C:"string"` - The default is /L literal. Explicitly combining the /L option with /C:"string" certainly works but is redundant.

`"string argument"` - The default depends on the content of the very first search string. (Remember that <space> is used to delimit search strings.) If the first search string is a valid regular expression that contains at least one un-escaped meta-character, then all search strings are treated as regular expressions. Otherwise all search strings are treated as literals. For example, "51.4 200" will be treated as two regular expressions because the first string contains an un-escaped dot, whereas "200 51.4" will be treated as two literals because the first string does not contain any meta-characters.

`/G:file` - The default depends on the content of the first non-empty line in the file. If the first search string is a valid regular expression that

contains at least one un-escaped meta-character, then all search strings are treated as regular expressions. Otherwise all search strings are treated as literals.

Recommendation - Always explicitly specify `/L` literal option or `/R` regular expression option when using "string argument" or `/G:file`.

Searching for spaces: when the search *string* contains multiple words, separated with spaces, then FINDSTR will return lines that contain either word (OR).
A literal search (/C:*string*) will reverse this behaviour and allow searching for a phrase or sentence. A literal search also allow searching for punctuation characters.

e.g. a text file `Demo.txt` contains the following

The quick brown fox
The really ^brown^ fox

Match the second line with: `FINDSTR /L /C:"^brown" Demo.txt`

FINDSTR Output

The format of matching line output from FINDSTR is:

filename:*lineNumber*:*lineOffset*:*text*

where

fileName = The name of the file containing the matching line. The file name is not printed if the request was explicitly for a single file, or if searching piped input or redirected input. When printed, the fileName will always include any path information provided. Additional path information will be added if the /S option is used. The printed path is always relative to the provided path, or relative to the current directory if none provided.

lineNumber = The line number of the matching line represented as a decimal value with 1 representing the 1st line of the input. Only printed if /N option is specified.

lineOffset = The decimal byte offset of the start of the matching line, with 0 representing the 1st character of the 1st line. Only printed if /O option is specified.

text = The binary representation of the matching line, including any <CR> and/or <LF>.

Nothing is left out of the binary output, such that this example that matches all lines will produce an exact binary copy of the original file:

```
FINDSTR "^" FILE >FILE_COPY
```

Using a script file

Multiple search criteria can be specified with a script file `/G`
Multiple FileNames to search can be specified with `/F`

When preparing a source or script file, place each filename or search criteria on a new line.
If several filenames are to be searched they must all exist or FINDSTR will fail with an error.

For example: to use the search criteria in Criteria.txt to search the files listed in Files.txt:

```
FINDSTR /g:Criteria.txt /f:Files.txt
```

Piping and Redirection

A text file can be piped or redirected into FINDSTR:

- Data stream from a pipe `TYPE file.txt | FINDSTR "searchString"`
- Stdin via redirection `FINDSTR "searchString" <file.txt`

The various data source specifications are mutually exclusive - FINDSTR can only work with one of the following: filename argument(s), /F:file option, redirected input, or piped input.

Piped and Redirected input may have <CR><LF> appended:

- If the input is piped in and the last character of the stream is not <LF>, then FINDSTR will automatically append <CR><LF> to the input. (XP, Vista and Windows 7.)
- If the input is redirected and the last character of the file is not <LF>, then FINDSTR will automatically append <CR><LF> to the input.

(Vista only), Note that in this case XP and Windows 7/2008 will **not** alter redirected input which may cause FINDSTR to hang indefinitely.

Errorlevel

FINDSTR sets `%ERRORLEVEL%` To 0 (False) if match is found in at least one line of at least one file.
FINDSTR sets `%ERRORLEVEL%` To 1 (True) if a match is not found in any line of any file.

```
Echo 12G6 |FindStr /R "[0-9]"
If %ERRORLEVEL% EQU 0 echo The string contains one or more numeric characters

Echo 12G6 |FindStr /R "[^0-9]"
If %ERRORLEVEL% EQU 0 echo The string contains one or more non numeric characters
```

Bugs

On XP and Windows 7. If the last character of a file used as redirected input does not end with <LF>, then FINDSTR will hang indefinitely once it reaches the end of the redirected file.

FINDSTR cannot search for null bytes commonly found in Unicode files.

Specifying multiple literal search strings can give unreliable results. The following FINDSTR example fails to find a match, even though it should:

```
echo ffffaaa|findstr /l "ffffaaa faffaffddd"
```

This bug has been confirmed on Windows Server 2003, Windows XP, Vista, and Windows 7.

Based on experiments, FINDSTR may fail if all of the following conditions are met:

- The search is using multiple literal search strings
- The search strings are of different lengths
- A short search string has some amount of overlap with a longer search string
- The search is case sensitive (no /I option)

It seems to always be the shorter search strings that fails, for more info see: FINDSTR fails to match multiple literal search strings

In early versions of `FindStr /F:file` a path length of more than 80 chars will be truncated.

Examples:

Search for "granny" OR "Smith" in the files Apples.txt or Pears.txt

```
FINDSTR "granny Smith" Apples.txt Pears.txt
```

Search for "granny Smith" in Contacts.txt (effectively the same as the FIND command)

```
FINDSTR /C:"granny Smith" Contacts.txt
```

Search every file in the current folder and all subfolders for the word "Smith", regardless of upper/lower case, note that `/S` will only search below the *current* directory:

```
FINDSTR /s /i smith *.*
```

Search all the text files in the current folder for the string "fiona", display the filenames in White on Green.

```
FINDSTR /A:2F /C:fiona *.txt
```

To find every line in novel.txt containing the word SMITH, preceeded by any number of spaces, and to prefix each line found with a consecutive number:

```
FINDSTR /b /n /c:" *smith" novel.txt
```

Finding a string only if surrounded by the standard delimiters
Find the word "computer", but not the words "supercomputer" or "computerise":

```
FINDSTR "\<computer\>" C:\work\inventory.txt
```

Find any words that begin with the letters 'comp', such as 'computerise' or 'compete'

```
FINDSTR "\<comp.*" C:\work\inventory.txt
```

Find any positive integers in the file sales.txt and include any lines that are a zero (0):

```
FINDSTR /r "^[1-9][0-9]*$ ^0$" Sales.txt
```

Credits:
Dave Benham - List of undocumented features and limitations of FINDSTR from StackOverflow

"Twenty years from now, you will be more disappointed by the things you didn't do than by the ones you did do. So throw off the bowlines, sail away from the safe harbour. Catch the trade winds in your sails. Explore. Dream. Discover" ~ Mark Twain

Related:

FINDSTR - Escapes and Length limits
FINDSTR - Searching across line breaks
FIND - Search for a text string in a file.
VBScript: Find and Replace
Powershell: Regular Expressions
Powershell: Where-Object - Filter objects passed along the pipeline.
Equivalent bash command (Linux): grep - Search file(s) for lines that match a given pattern