/T=*t*	Sets the trace mode. *t*=0: Lines and small square markers (default). *t*=1: User settings unchanged.
/W=*winName*	Edits traces in the named graph window or subwindow. When omitted, action will affect the active window or subwindow. This must be the first flag specified when used in a Proc or Macro or on the command line. When identifying a subwindow with *winName*, see **Subwindow Syntax** on page III-92 for details on forming the window hierarchy.

Details

The GraphWaveEdit operation is not multidimensional aware. See **Analysis on Multidimensional Waves** on page II-95 for details.

See Also

The **GraphNormal**, **GraphWaveDraw** and **DrawAction** operations.

Trace Names on page II-282, **Programming With Trace Names** on page IV-87.

Grep

```
Grep [flags][srcFileStr ][srcTextWaveName][ as [destFileOrFolderStr]
    [destTextWaveName]]
```

The Grep operation copies lines matching a search expression from a file on disk, the Clipboard, or rows from a text wave to a new or existing file, an existing text wave, the History area, the Clipboard, or to S_value as a string list.

Source Parameters

The optional *srcFileStr* can be

- The full path to the file to copy lines from (in which case /P is not needed).
- The partial path relative to the folder associated with *pathName.*
- The name of a file in the folder associated with *pathName.*
- "Clipboard" to read lines of text from the Clipboard (in which case /P is ignored).

If Igor can not determine the location of the source file from *srcFileStr* and *pathName*, it displays a dialog allowing you to specify the source file.

The optional *srcTextWaveName* is the name or path to a text wave.

Only one of *srcFileStr* or *srcTextWaveName* may be specified. If neither is specified then an Open File dialog is presented allowing you to specify a source file.

Destination Parameters

The optional *destFileOrFolderStr* can be

- The name of (or path to) an existing folder when /D is specified.
- The name of (or path to) a possibly existing file.
- "Clipboard", in which case the matching lines are copied to the Clipboard (and /P and /D are ignored). The text can be retrieved with the **GetScrapText** function.

If *destFileOrFolderStr* is a partial path, it is relative to the folder associated with *pathName*.

If /D is specified, the source file is created inside the folder using the source file name.

If Igor can not determine the location of the destination file from *pathName*, *srcFileStr*, and *destFileOrFolderStr*, it displays a Save File dialog allowing you to specify the destination file (and folder).

The optional *destTextWaveName* is the name or path to an existing text wave. It may be the same wave as *srcTextWaveName*.

Only one of *destFileOrFolderStr* or *destTextWaveName* may be specified.

If no destination file or text wave is specified then matching lines are printed in the history area, unless the /Q flag is specified, in which case the matching lines aren't printed or copied anywhere (though the output variables are still set).

Use /LIST to set S_value to a string list containing the matching lines or rows.

Use /INDX to create a wave W_index containing the zero-based row or line number where matches were found.

Parameter Details

If you use a full or partial path for either *srcFileStr* or *destFileOrFolderStr*, see **Path Separators** on page III-451 for details on forming the path.

Folder paths should not end with single path separators. See **MoveFolder**'s Details section.

Flags

/A Appends matching lines to the destination file, creating it if necessary, appends text to the Clipboard if the *destFileOrFolderStr* is "Clipboard", or appends rows to the destination text wave. Has no effect on output to the History area.

/D Interprets *destFileOrFolderStr* as the name of (or path to) an existing folder (or directory). Without /D, *destFileOrFolderStr* is the name of (or path to) a file. It is ignored if the destination is a text wave, the Clipboard, or the History area.

If *destFileOrFolderStr* is not a full path to a folder, it is relative to the folder associated with *pathName*.

/DCOL={*colNum*} Useful only when the destination is *destTextWaveName*.

Copies matching lines of text from the source file, Clipboard, or *srcTextWaveName* to column *colNum* of *destTextWaveName*, with any terminator characters removed.

The default when the source is a file or the Clipboard is `/DCOL={0}`, which copies matching lines into the first column of *destTextWaveName*.

The default when the source is *srcTextWaveName* is to copy each column of a matched row to the corresponding column in *destTextWaveName*.

/DCOL must be used with the /A flag, otherwise the destination wave will have only 1 column.

/DCOL={[*colNum*] [, *delimStr*], …}

Useful only when the source is *srcTextWaveName* and the destination is a file, the Clipboard, History area, or S_value.

Copies multiple columns in any order from matching rows of *srcTextWaveName* to the destination file, Clipboard, History area, or S_value.

Construct the line by appending the contents of the numbered column and the *delimStr* parameters in the order specified. The output line is terminated as described in **Line Termination**.

/E=*regExprStr* Specifies the Perl-compatible regular expression string. A line must match the regular expression to be copied to the output file. See **Regular Expressions**.

Multiple /E flags may be specified, in which case a line is copied only if it matches every regular expression.

/E={*regExprStr*, *reverse*}

Specifies the Perl-compatible regular expression string, *regExprStr*, for which the sense of the match can be changed by *reverse*.

reverse=1: Matching expressions are taken to **not** match, and vice versa. For example, use `/E={"CheckBox",1}` to list all lines that do not contain "CheckBox".

reverse=0: Same as /E=*regExptrStr*.

/ENCG=*textEncoding*

Specifies the text encoding of named text file. This flag applies if the source is a file and is ignored if the source is the clipboard or a text wave.

See **Text Encoding Names and Codes** on page III-490 for a list of accepted values for *textEncoding*.

If you omit /ENCG, Grep uses the default text encoding as specified by the Misc→Text Encoding→Default Text Encoding menu.

Passing 0 for textEncoding acts as if /ENCG were omitted. Passing 255 (binary) for textEncoding is treated as an error because the source text must be internally converted to UTF-8 and there is no valid conversion from binary to UTF-8.

/GCOL=*grepCol* Greps the specified column of *srcTextWaveName*, which is a two-dimensional text wave. The default search is on the first column (`grepCol=0`). Use `grepCol=-1` to match against any column of *srcTextWaveName*.

Does not apply if the source is a file or Clipboard.

/I Requires interactive searching even if *srcFileStr* and *destFileOrFolderStr* are specified and if the source file exists. Same as /I=3.

/I=*i* Specifies the degree of file search interactivity with the user.

i=0: Default; interactive only if *srcFileStr* is not specified or if the source file is missing. Same as if /I were not specified.

Note: This is different behavior than other commands such as **CopyFile**.

i=1: Interactive even if *srcFileStr* is specified and the source file exists.

i=2: Interactive even if *destFileOrFolderStr* is specified.

i=3: Interactive even if *srcFileStr* is specified, the source file exists, and *destFileOrFolderStr* is specified.

/INDX Creates in the current data folder an output wave W_Index containing the line numbers (or row numbers) where matching lines were found. If this is the only output you need, also use the /Q flag.

/LIST[=*listSepStr*] Creates an output string variable S_value containing a semicolon-separated list of the matching lines. If *listSepStr* is specified, then it is used to separate the list items. See **StringFromList** for details on string lists. If this is the only output you need, also use the /Q flag.

/M=*messageStr* Specifies the prompt message in any Open File dialog. If /S is not specified, then *messageStr* will be used for both the Open File and Save File dialogs.

/O Overwrites any existing destination file.

/P=*pathName* Specifies the folder containing the source file or the folder into which the file is copied. *pathName* is the name of an existing symbolic path.

Both *srcFileStr* and *destFileOrFolderStr* must be either simple file or folder names, or paths relative to the folder specified by *pathName*.

/Q Prevents printing results to an output file, text wave, History, or Clipboard. Use /Q to check for a match to *regExprStr* by testing the value of V_flag, V_value, S_value (/LIST), or W_Index (/INDX) without generating any other matching line output.

Note: When using /Q neither *destFileOrFolderStr* nor *destTextWaveName* may be specified.

/S=*saveMessageStr* Specifies the prompt message in any Save File dialog.

/T=*termCharStr* Specifies the terminator character.

`/T=(num2char(13))`	Carriage return (CR, ASCII code 13).
`/T=(num2char(10))`	Linefeed (LF, ASCII code 10).
`/T=";"`	Semicolon.
`/T=""`	Null (ASCII code 0).

See **Line Termination** for the default behavior of the terminator character.

/Z[=*z*] Prevents aborting of procedure execution when attempting to open a nonexistent file for searching. Use /Z if you want to handle this case in your procedures rather than having execution abort.

z=0:	Same as no /Z at all.
z=1:	Open file only if it exists. `/Z` alone is the same as `/Z=1`.
z=2:	Open file if it exists and display a dialog if it does not exist.

Line Termination

Line termination applies mostly to source and destination files. (The Clipboard and history area delimit lines with CR, and text waves have no line terminators.)

If /T is omitted, Grep will break file lines on any of the following: CR, LF, CRLF, LFCR. (Most Macintosh files use CR. Most Windows files use CRLF. Most UNIX files use LF. LFCR is an invalid terminator but some buggy programs generate files that use it.)

Grep reads whichever of these terminator(s) appear in the source file and use them to write lines to any output file.

The terminator(s) are removed before the line is matched against the regular expression.

For lines that match *regExprStr*, terminator(s) in the input file are transferred to the output file unless the output is the Clipboard or history area, in which case the output terminator is always only CR (like **LoadWave**). This means you can transparently handle files that use CR, LF, CRLF, or LFCR as the terminator, and omitting /T will be suitable for most cases.

If you use the /T flag, then Grep will terminate line-from-file reads on the specified character only and will output the specified character into any output file.

"Lines" in One-dimensional Text Waves

Grep considers each row of *srcTextWaveName* or *destTextWaveName* to be a "line" of input or output.

When the destination is a file, the Clipboard, or the History area, Grep copies all of the text in a matching row of *srcTextWaveName* to the file and terminates the line. See **Line Termination** for the rules on line terminators.

When the destination is a *destTextWaveName*, Grep simply copies all the text in a matching row to a row in *destTextWaveName*, without adding or omitting any terminators.

"Lines" and Columns in Two-Dimensional Text Waves

Grep by default *matches* against only the first column (column 0) of each row of *srcTextWaveName*. You can use the `/GCOL=grepCol` flag to specify a different column to match against. Use `/GCOL=-1` to match against any column of *srcTextWaveName*.

When the source is a text wave and the destination is a file, the Clipboard, or the History area, Grep by default *copies* only the first column (column 0) to the destination.

Use the `/DCOL={colNum1, delimStr1, colNum2, delimStr2,...colNumN}` to print multiple columns (in any order) with delimiters after each column (the last column number need not be followed by a delimiter string). The output line is terminated with CR or *termcharStr* as described in **Line Termination**.

When both the source and destination are text waves and append (/A) is *not* specified, the destination text wave is redimensioned to have the same number of columns as the source text wave, and all columns of matching rows of *srcTextWaveName* are copied to *destTextWaveName*.

When both the source and destination are text waves and append /A is specified, then the number of columns in *destTextWaveName* is left unchanged, and each column of *srcTextWaveName* is copied to the corresponding column of *destTextWaveName*.

If the destination is a text wave and the source is a file or the Clipboard, each line (without the terminator) is copied to the first column of the destination text wave, or use /DCOL={*destColNum*} to put the text into a different column.

Output Variables

The Grep operation returns information in the following variables. When running in a user-defined function these are created as local variables. Otherwise they are created as global variables in the current data folder.

V_flag	0: Output successfully generated. -1: User cancelled either the Open File or Save File dialogs. Other: An error occurred, such as the specified file does not exist.
V_value	The number of input lines that matched the regular expression.
V_startParagraph	Zero-based line number into the file or Clipboard (or the row number of a source text wave) where the first regular expression was matched. Also see the **/INDX** flag.
S_fileName	Full path to the source file, the source text wave, or "Clipboard". If an error occurred or if the user cancelled, it is an empty string.
S_path	Full path to the destination file or destination text wave. "Clipboard": If *destFileOrFolderStr* was the Clipboard. "History": If the output was printed to the history area of the window. "": If an error occurred, if the user cancelled, or if /Q was specified.
S_value	Contains matching lines as a string list only if /LIST is specified.

Regular Expressions

A regular expression is a pattern that is matched against a subject string from left to right. Most characters stand for themselves in a pattern, and match the corresponding characters in the "subject".

In the case of Grep, the "subject" is each line of the source file or Clipboard, or each row in the source text wave.

The regular expression syntax supported by Grep, **GrepList**, and **GrepString** is based on the "Perl-Compatible Regular Expression" (PCRE) library.

The syntax is similar to regular expressions supported by various UNIX and POSIX egrep(1) commands. See **Regular Expressions** on page IV-176 for more details.

As a trivial example, the pattern "Fred" as specified here:

```
Grep/P=myPath/E="Fred" "afile.txt" as "FredFile.txt"
```

matches lines that contain the string "Fred" anywhere on the line.

Character matching is *case-sensitive* by default, similar to strsearch. Prepend the Perl 5 modifier (?i) to match upper and lower-case versions of "Fred":

```
// Copy lines that contain "Fred", "fred", "FRED", "fREd", etc
Grep/P=myPath/E="(?i)fred" "afile.txt" as "AnyFredFile.txt"
```

To copy lines that do *not* match the regular expression, set the /E flag's reverse parameter:

```
// Copy lines that do NOT contain "Fred", "fred", "fREd", etc.
Grep/P=myPath/E={"(?i)fred",1} "afile.txt" as "NotFredFile.txt"
```

Note: Igor doesn't use the opening and closing regular expression delimiters that UNIX grep or Perl use: they would have used "/Fred/" and "/(?i)fred/".

Regular expressions in Igor support the expected metacharacters and character classes that make the whole grep paradigm so useful. For example:

```
// Copy lines that START with a space or tab character
Grep/P=myPath/E="^[ \\t]" "afile.txt" as "LeadingTabsFile.txt"
```

For a complete description of regular expressions, see **Regular Expressions** on page IV-176, especially for a description of the many uses of the regular expression backslash character (see **Backslash in Regular Expressions** on page IV-179).

Note: Because Igor Pro also has special uses for backslash (see **Escape Sequences in Strings** on page IV-14), you must double the number of backslashes you would normally use for a Perl or grep pattern. Each pair of backslashes identifies a single backslash for the Grep command.

For example, to copy lines that contain "\z", the Perl pattern would be `\\z`, but the equivalent Grep expression would be `/E="\\\\z"`.

See **Backslash in Regular Expressions** on page IV-179 for a more complete description of backslash behavior in Igor Pro.

Examples

```
// Copy lines in afile.txt containing "Fred" (case sensitive)
// to an output file named "AnyFredFile.txt" in the same directory.
Grep/P=myPath/E="Fred" "afile.txt" as "AnyFredFile.txt"

// Copy lines in afile.txt containing "Fred" and "Wilma" (case-insensitive)
// to a text wave (which must exist andis overwritten):
Make/O/N=0/T outputTextWave
Grep/P=myPath/E="(?i)fred"/E="(?i)wilma" "afile.txt" as outputTextWave

// Print lines in afile.txt containing "Fred" and "Wilma" (case-insensitive)
// to the history area
Make/O/N=0/T outputTextWave
Grep/P=myPath/E="(?i)fred"/E="(?i)wilma" "afile.txt"

// Test whether afile.txt contains the word "boondoggle", and if so,
// on which line the first occurence was found, WITHOUT creating any output.
//
// Note: the \\b sequences limit matches to a word boundary before and after
// "boondoggle", so "boondoggles" and "aboondoggle" won't match.
//
Grep/P=myPath/Q/E="(?i)\\bBoondoggle\\b" "afile.txt"
if( V_value )                       // at least one instance was found
   Print "First instance of \"boondoggle\" was found on line", V_startParagraph
endif

// Create in S_value a string list of the lines as \r - separated list items:
Grep/P=myPath/LIST="\r"/Q/E="(?i)\\bBoondoggle\\b" "afile.txt"
if( V_Value )                       // some were found
   Print S_value
endif

// Create in W_index a list of the 0-based line numbers where "boondoggle"
// or "boondoggles", etc was found in afile.txt.
Grep/P=myPath/INDX/Q/E="(?i)boondoggle" "afile.txt"
if( V_flag == 0 )   // grep succeeded, perhaps none were found; let's see where
   WAVE W_Index     // needed if in a function
   Edit W_Index     // show line numbers in a table.
endif

// (Create a string list and text wave for the following examples.)
String list= CTabList()           // "Grays;Rainbow;YellowHot;..."
Variable items= ItemsInList(list)
Make/O/T/N=(items) textWave= StringFromList(p,list)

// Copy rows of textWave that contain "Red" (case sensitive)
// to the Clipboard as carriage-return separated lines.
Grep/E="Red" textWave as "Clipboard"

// Copy lines of the Clipboard that do NOT contain "Blue"
// (case in-sensitve) back to the Clipboard, overwriting what was there:
Grep/E={"(?i)blue",1} "Clipboard" as "Clipboard"
```