Contents

Introduction

This program contains a set of common UNIX filters. These process an input file or files, optionally apply various transformations or conversions, and write the results back to the original file(s) or append them to a single output file. The following filters are provided in this version of the product:

- cat(1). Concatenates one or more input files and appends them to a single output file, which is created if it does not already exist.
- col(1). Filters reverse line-feeds and various other control characters. Optionally, this filter can also be used to remove backspace sequences or to convert spaces to tabs.
- expand(1). Performs the opposite function to col(1) and expands tabs to spaces.
- fmt(1). A simple text processing utility for performing modest tasks such as formatting mail messages. Text can be formatted using block, indented, crown or centered paragraphs, optionally with left and right text justification.
- fold(1). Folds lines from its input files, breaking the lines to have a maximum column width (after backspace and tab processing) or a maximum number of bytes. Optionally, lines can be broken at the last blank character within the specified column or byte width.
- grex(1). Is a simple stream editor that performs various ed(1)-like editing commands in place. These commands include text substitution or deletion, line deletion, and multi-line text insertion.
- nl(1). Numbers the lines of one or more text files under the control of various formatting options. Numbering can be left or right justified, and can be padded up to a specified width with spaces or zeros.
- sort(1). Sorts and/or merges one or more text files according to a sort key specification and a series of comparison options. Comparisons can be numeric or lexicographic and the order of sorting can be reversed.
- tr(1). Substitutes or deletes specified characters in one or more input files. This is a general purpose fiter with all manner of uses; for example, converting all lower-case to upper-case characters (or vice versa), deleting control characters from the input stream, replacing multiple adjacent occurrences of a a character with a single occurrence.

See Also

Cat

Name

cat - concatenate files

Description

The Cat utility reads files in sequence and appends their contents to a specifed output file, which is created if it does not already exist. Because Cat reads files block-by-block, it can be used to concatenate any type of file (not just text files).

Operands

Input Files

One or more filenames (possibly containing DOS wild-card characters) naming the input file or files to be processed. Multiple filenames appearing in this field should be separated by a space-character (e.g. "foo.bar *.cpp *.h"). A filename can be added by typing its name or by double-clicking a filename in the associated directory listbox. The current drive/directory can be changed by double-clicking the appropriate drive/directory name in the directory listbox.

Output File

A filename identifying the output file. The input files will be appended to this file, preserving its original contents.

IMPORTANT!!! The output filename must NOT be the same as any of the input filenames. If it is, the input file will be destroyed and the behaviour of this utility is unspecified.

Col

Name

col - filter reverse line-feeds

Description

The Col utility reads from the specified input text files and filters line motions in place or appends the results to a single output file. It performs the line overlays implied by reverse line-feeds, and by forward and reverse half-line feeds. Unless the "No space to tab conversion option" is checked, all blank characters in the input will be converted to tab characters wherever possible.

On input, the only control characters recognised are space, backspace, tab, carriage-return and newline characters, SI, SO, VT, reverse line-feed, forward half line-feed and reverse half line-feed. The only other characters copied to the output are those that are printable.

Operands

Input Files:

One or more filenames (possibly containing DOS wild-card characters) naming the input file or files to be processed. Multiple filenames appearing in this field should be separated by a space-character (e.g. "foo.bar *.cpp *.h"). A filename can be added by typing its name or by double-clicking a filename in the associated directory listbox. The current drive/directory can be changed by double-clicking the appropriate drive/directory name in the directory listbox.

Output File (Optional):

A filename identifying the output file. The input files will be appended to this file (if specified), preserving its original contents. If this field is blank, the processed output will be written back to the original input file(s).

Flags

- Filter backspaces. If checked, backspace sequences are processed such that where two or more characters would appear in the same place on a display device, only the last one read will be output.
- Allow forward half-line feeds. If checked, suppresses the normal treatment of half-line motions as described above.
- No space to tab conversion. If checked, prevents Col from converting space characters to tab characters. Tab stops are considered to be at every 8th column position.
- Ignore unknown escapes. If checked, forces unrecognised escape sequences to be passed through unchanged. Normally, Col will remove any escape sequences found in its input that are not specified above.

Expand

Name

expand - convert tabs to spaces

Description

The Expand utility replaces text files or appends all output to a single text file with tab characters replaced by one or more space characters needed to pad to the next tab stop. Any backspace or carriage return characters will be copied to the output and cause the column position count for tab stop calculations to be decremented.

Operands

Input Files:

One or more filenames (possibly containing DOS wild-card characters) naming the input file or files to be processed. Multiple filenames appearing in this field should be separated by a space-character (e.g. "foo.bar *.cpp *.h"). A filename can be added by typing its name or by double-clicking a filename in the associated directory listbox. The current drive/directory can be changed by double-clicking the appropriate drive/directory name in the directory listbox.

Output File (Optional):

A filename identifying the output file. The input files will be appended to this file (if specified), preserving its original contents. If this field is blank, the processed output will be written back to the original input file(s).

Tab Length:

Specifes tab stop positions as a value N; that is, tab stops are assumed to be set at every N columns.

Fmt

Name

fmt - simple text formatter

Description

Fmt is a text processing utility intended for simple tasks such as formatting mail messages. It processes its input text files a line at a time, joining lines to form paragraphs of a similar type and with similar alignment. Paragraphs can be block, indented, crown or centered, optionally with left and right text justification. For example:

```
The Moon shone on the village Green,
It also shone on Nell.
Was she picking daisies?
Was she bloody hell.
```

becomes:

```
The  Moon shone on the village Green, It
also  shone  on  Nell.  Was  she picking
daisies? Was she bloody hell.
```

Using a line width of 40, block paragraphs, and text justification. Using centered paragraphs, it becomes:

```
  The Moon shone on the village Green,
         It also shone on Nell.
        Was she picking daisies?
          Was she bloody hell.
```

Paragraphs are delimited by blank lines, changes in indentation and lines starting with a dot (.). The latter is so that nroff/troff source files can be preserved.

Operands

Input Files:

One or more filenames (possibly containing DOS wild-card characters) naming the input file or files to be processed. Multiple filenames appearing in this field should be separated by a space-character (e.g. "foo.bar *.cpp *.h"). A filename can be added by typing its name or by double-clicking a filename in the associated directory listbox. The current drive/directory can be changed by double-clicking the appropriate drive/directory name in the directory listbox.

Output File (Optional):

A filename identifying the output file. The input files will be appended to this file (if specified), preserving its original contents. If this field is blank, the processed output will be written back to the original input file(s).

Line Width:

Sets the maximum line length to the specified number of characters (default 72). Output text lines will be broken at the last blank character before this number and justified (if Justify Text is checked).

Justify Text:

Produces a justified right margin by inserting extra blanks into output lines as necessary. Note that

spaces are preserved in the input, so repeated uses of Fmt on the same text may eventually produce strange looking output. To overcome this problem, replace multi-space sequences in the input with a single space by using the Squeeze option of Tr before calling Fmt.

Paragraph Type:

Specifies the output paragraph style as follows:

- Block. Output paragraphs are uniformly indented, including the first line.
- Indented. The first line of each output paragraph is indented by 8 characters; all other lines are uniformly indented.
- Crown. The first line is negatively indented by 8 characters; all other lines are uniformly indented.
- Centered. Input lines are centered in the output. When this type of paragraph is selected, input indentation is ignored and input lines are not joined.

Fold

Name

fold - filter for folding lines

Description

The Fold utility will fold lines from its input text files, breaking the lines to have a maximum width column positions (or bytes, see below). Lines will be broken by the insertion of a newline character, such that each output line is the maximum width possible that does not exceed the specifed number of column positions (or bytes).

Carriage-return, backspace or tab characters encountered in the input (if bytes is not specifed) will cause column position calculations to be adjusted accordingly. Tab stops are assumed to be ever 8 column positions.

Operands

Input Files:

One or more filenames (possibly containing DOS wild-card characters) naming the input file or files to be processed. Multiple filenames appearing in this field should be separated by a space-character (e.g. "foo.bar *.cpp *.h"). A filename can be added by typing its name or by double-clicking a filename in the associated directory listbox. The current drive/directory can be changed by double-clicking the appropriate drive/directory name in the directory listbox.

Output File (Optional):

A filename identifying the output file. The input files will be appended to this file (if specified), preserving its original contents. If this field is blank, the processed output will be written back to the original input file(s).

Units per Line:

Specifies the maximum line length calculated in column positions or bytes, depending on which "Units" are selected (see below).

Break at White-space:

If a line contains a blank character (space or tab) within the first width column positions (or bytes), the line will be broken after the last blank character rather than at the specified column (byte) position. This is useful for preserving whole words in natural language text files.

Units:

- Columns. Specifies that line widths are to be calculated in column positions (that is, after carriage return, tab and backspace processing).
- Bytes. Specifies that line widths are to be calculated in bytes.

Grex

Name

grex - simple stream editor

Description

The Grex utility is a simple stream editor that reads one or more text files, makes editing changes according to its operands, and writes the results back to the original files or appends them to a single output file.

Operands

Input Files:

One or more filenames (possibly containing DOS wild-card characters) naming the input file or files to be processed. Multiple filenames appearing in this field should be separated by a space-character (e.g. "foo.bar *.cpp *.h"). A filename can be added by typing its name or by double-clicking a filename in the associated directory listbox. The current drive/directory can be changed by double-clicking the appropriate drive/directory name in the directory listbox.

Output File (Optional):

A filename identifying the output file. The input files will be appended to this file (if specified), preserving its original contents. If this field is blank, the processed output will be written back to the original input file(s).

Search String:

Specifies a regular expression which Grex uses to match lines and items of text within lines. Editing "Actions" are only performed on matched input lines; whether the whole line or only the matched text segment is affected depends on the selected Action (see below).

Regular Expression

Replacement:

If the selected Action is "Substitute", this field contains a plain text string that replaces each instance of matched text. If the Action is "Append line" or "Insert line", this field contains either a single text line to be appended/inserted, or the name of a file containing the text prefixed by the commercial-at character (@); for example, "@insert.txt" directs Grex to read insert/append lines from the file insert.txt. This field is ignored for all other Actions.

Action:

- Substitute. Specifies that all matched occurrences of the Search String are to be replaced by the contents of Replacement, this includes multiple occurrences on a single text line.
- Delete string. Specifies that all matched occurrences of the Search String are to be deleted.
- Delete line. Specifies that all lines containing a matching occurrence of the Search String are to be deleted.
- Append line(s). Specifies that the line or lines of text specified by Replacement are to be added to the output AFTER each line containing a matching occurrence of the Search String.
- Insert line(s). Specifies that the line or lines of text specified by Replacement are to be added to the output BEFORE each line containing a matching occurrence of the Search String.

Regular Expression

A regular expression is made up of ordinary characters and metacharacters. An ordinary character matches itself; for example, the re "aardvark" matches the character sequence <a><a><r><d><v><a><r><k> anywhere in an input line. A metacharacter or metacharacter sequence has special meaning as described below.

Metacharacters

. A dot (.) matches any character. For example, the re "ab." matches the character sequence <a><b><any-character>.

* An asterisk (*) matches the previous ordinary character, metacharacter or metacharacter sequence any number of times. For example, the re "ab.*" matches the character sequence <a><b><any-character...>.

\ A backslash (\) escapes the following character and is commonly used to escape the meaning of a metacharacter. For example, the re "ab\." matches the character sequence <a><b><.>.

\n This expression matches the same string of characters as was matched by the nth expression enclosed between the metacharacter sequence \(and \) (see below for details), appearing earlier in the same regular expression. For example, the re "^\(abc\)\1$" matches a line consisting of two repeated appearances of the string <a><b><c>.

^ A caret (^) anchors a regular expression at the start of an input line. For example, the re "^abc" will only match input lines that start with the character sequence <a><b><c>

$ A dollar ($) anchors a regular expression at the end of an input line. For example, the re "abc$" will only match input lines that end with the character sequence <a><b><c>.

Metacharacter Sequences

[...] A bracketed re is a metacharacter sequence that defines a scanset. A scanset defines single characters or character ranges that are matched against the next character in the input line. For example, the re "^[abc]" will match any input line that begins with the characters <a>, <b> or <c>. A character range is denoted by dash (-) separated character sequences; that is, the re "^[a-z]" matches any line beginning with a lower-case letter, the re "^[a-zA-Z]" matches any line beginning with a lower-case or an upper-case letter. Within scansets, metacharacters, other than a dash, lose their special meaning. A dash character appearing as the first or last character in a scanset loses its special meaning and matches itself. A caret (^) appearing as the first character in a scanset causes the scanset to be complemented; that is, the re will match characters NOT specified in the scanset.

\{m,n\} Is a range expression that matches the previous ordinary character or metacharacter a specified number of times. The values of m and n must be integers in the range 1-255. The expression \{m\} matches exactly m occurrences; \{m,\} matches at least m occurrences; \{m,n\} matches any number of occurrences between m and n inclusive. Wherever a choice exists, the re matches as many occurrences as possible.

\(re\) An re enclosed between \(and \) matches whatever the unadorned re matches. However, the enclosed re can also be used for repeating a match (see \n above).

Nl

Name

nl - line numbering filter

Description

The nl utility reads lines from the input text file or files, adds line numbers on the left, and writes the output back to the original files, or concatenates all the output in a single file. Lines are numbered under the control of various options, which permit the numbering to be left or right justified, with leading spaces or zeros. The starting number, increment, width and the character or characters separating the number from the rest of the text can all be controlled.

Operands

Input Files:

One or more filenames (possibly containing DOS wild-card characters) naming the input file or files to be processed. Multiple filenames appearing in this field should be separated by a space-character (e.g. "foo.bar *.cpp *.h"). A filename can be added by typing its name or by double-clicking a filename in the associated directory listbox. The current drive/directory can be changed by double-clicking the appropriate drive/directory name in the directory listbox.

Output File (Optional):

A filename identifying the output file. The input files will be appended to this file (if specified), preserving its original contents. If this field is blank, the processed output will be written back to the original input file(s).

Type

- Number all lines. All lines in the input file(s) will be numbered, regardless of their content
- Number non-blank lines only. Lines containing only space characters (space, tab and newline) will not be numbered in the output. All other lines will be numbered as above.

Format

- Right justified. Line numbers will right-justified with leading spaces.
- Left justified. Line numbers will be left-justified with trailing spaces.
- Leading zeros. Line numbers will be right-justified with leading zeros.

Options

- First line number. Gives the number to be attached to the first line of text in the output (default = 1).
- Increment. Gives the number by which successive line numbers will be incremented (default = 1).
- Width (chars). Gives the width of a line number in characters (default = 6).
- Separator. Gives the character or characters to be used to separate the line number from the text of each input line. This can contain any printable character, plus the special characters '\t' (tab) and '\n' (newline). The default is a single tab character.

Sort

Name

sort - sort or merge text files

Description

The sort utility performs one of the following functions

- Sorts lines in one or more text files individually and either writes the results back to the original files, or appends the sorted output to a single named file
- Sorts and merges one or more text files and writes the results to a named output file, which may be one of the input files

Comparisons are based on a sort key extracted from each input line, or the entire line if no sort key is specified.

Operands

Input Files:

One or more filenames (possibly containing DOS wild-card characters) naming the input file or files to be processed. Multiple filenames appearing in this field should be separated by a space-character (e.g. "foo.bar *.cpp *.h"). A filename can be added by typing its name or by double-clicking a filename in the associated directory listbox. The current drive/directory can be changed by double-clicking the appropriate drive/directory name in the directory listbox.

Output File:

For Action = Sort only, this field is optional. If it is left blank, each input file will be sorted and the results will overwrite the original contents of each file. Otherwise, if a filename is specified in this field, each input file will be sorted and the results will be appended to this file

For Action = Sort and merge, this field is mandatory. The sorted and merged ouput will be written to the named file, overwriting its original contents (if any). The output file can be one of the input files.

Action:

- Sort only. Indicates that each input file should be sorted separately.
- Sort and merge. Indicates that the input files should be concatenated and sorted as a whole.

Restricted Sort Key:

These two fields allow a restricted sort key to be specified, which limits comparisons to a specified part of each input line.

- Sort key. This gives the field numbers, and optionally character positions, where the sort key will start and end, in the form:

  startfield[.startchar][,endfield[.endchar]]

  Fields within lines, and characters within fields, begin numbering at 1. "startfield" and "startchar" give the starting field and character position for the sort key; if "startchar" is omitted, 1 is assumed. "endfield" and "endchar" give the terminating field and character position. If both are omitted, the sort key is assumed to extend to the end of each input line; if "endchar" is omitted, the last character of the sort key is assumed to be the last character of "endfield".

For example, sort key "1.2,5.4" defines that the sort key will begin at field 1 character 2, and extend to field 5 character 4. Sort key "1,1" says that the entire first field will act as the sort key. Sort key "5" defines that the sort key will start at field 5 and extend to the end of the input line.

If no sort key is specified, comparisons will start at field "1.1" (that is, field number 1, character position 1), and extend to the end of each input line.

- Field separator. This identifies the single character to be used as the field separator character. Each occurrence of the separator character is significant; for example, <char><char> delimits an empty field. If no field separator is specified, blank characters (space and tab) are used as default field separators. In this case, each maximal non-empty sequence of blanks that follows a non-blank character is a field separator.

Flags:

- Dictionary ordering. Specifies that only blank and alphanumeric characters are significant in comparisons.
- Fold character case. Consider all lower-case characters as upper-case characters for comparison purposes.
- Sort numerically. Restrict the sort key to an initial numeric string, consisting of optional blank characters, an optional minus sign, and zero or more digits with an optional decimal-point character. Otherwise comparisons are performed lexicographically.
- Reverse sort order. Reverse the sense of comparisons (i.e., sort last to first).
- Ignore non-printing characters. Ignore all characters that are non-printable, according to character type information in the C program locale (e.g., ASCII control characters).
- Ignore leading blanks. Discard leading blanks when determining the sort key.
- Uniq. When writing the sorted output, suppress multiple instances of lines that have equal sort keys.

Tr

Name

tr - translate characters

Description

The Tr utility replaces the input files in place, or appends them to a single output file, with translation, deletion or squeezing of selected characters. The String1 and String2 fields control which translations will occur while copying characters as specified under Operands below.

Operands

Input Files:

One or more filenames (possibly containing DOS wild-card characters) naming the input file or files to be processed. Multiple filenames appearing in this field should be separated by a space-character (e.g. "foo.bar *.cpp *.h"). A filename can be added by typing its name or by double-clicking a filename in the associated directory listbox. The current drive/directory can be changed by double-clicking the appropriate drive/directory name in the directory listbox.

Output File (Optional):

A filename identifying the output file. The input files will be appended to this file (if specified), preserving its original contents. If this field is blank, the processed output will be written back to the original input file(s).

String1:

Specifes the character or characters to be actioned, which can be specified in one of the following forms:

- c. Any single character not described by one of the following conventions represents itself.
- \c. The backslash escape sequences \a, \b, \f, \n, \r, \t, \v represent the ASCII control characters alert, backspace, form-feed, newline, carriage-return, tab and vertical-tab respectively. Any other character following a backslash character represents itself (including a backslash).
- \0nnn. Specifies an octal sequence representing a character value in the range 0 to 255
- [c1-c2]. Specifies a range of character values from "c1" to "c2", where "c1" and "c2" can be specified in any of the forms identified above. Note that unlike scansets in regular expressions, each range in Tr must be specifed withing its own set of enclosing square brackets. Thus "[A-Z][a-z]" specifies all English character values, not "[A-Za-z]".

String1 can contain multiple translation characters. For example, "[A-Z]\n\t" specifies all the upper-case English characters and the control characters newline and tab. This field is mandatory for all settings of Action.

String2:

This field is only required when Action is set to "Translate". Each character value in String2 replaces the corresponding character value in String1 in the output. All character specification forms identified above are valid in String2, plus the following additional form is added:

- [c1*n]. Represents "n" repeated occurrences of the character "c1". If "n" is omitted or zero, it is interpreted as being large enough to extend the String2-based sequence to the length of the String1-based sequence. "n" is interpreted as a decimal value.

Note that the length of String2 must always equal the length of String1, otherwise a string error is

indicated and no translations will occur.

Complement String1:

If checked, this indicates that the set of character values specified by String1 should be complemented, within the 8-bit domain 0-255. Thus a String1 setting of "[A-Z]" indicating all upper-case English characters, becomes all characters other than the upper-case English characters when complemented.

Action:

- Substitute. Specifies that all character values indicated in String1 are to be replaced by corresponding character values from String2 in the output.
- Delete. Specifies that all character values indicated in String1 are to be deleted from the output. String2 is ignored.
- Squeeze. Specifies that repeated occurrences of String1 character values in the input are to be replaced by a single occurrence of that character value in the output. String2 is ignored.