gnu:guide/textutils

COLLABORATORS

	TITLE : gnu:guide/textutils	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		April 16, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

gnu:guide/textutils

1.1 gnu:guide/textutils.guide

```
                GNU text utilities
******************

   This manual minimally documents version GNU textutils 1.11 of the
GNU text utilities.



                Introduction
                                        Caveats, overview, and authors.

                Common options
                                      Common options.

                Output of entire files
                   cat tac nl od

                Formatting file contents
                            fmt pr fold

                Output of parts of files
                   head tail split csplit

                Summarizing files
                        wc sum cksum

                Operating on sorted files
                   sort uniq comm

                Operating on fields within a line
                 cut paste join

                Operating on characters
                   tr expand unexpand

                Index
                          General index.
```

1.2 textutils.guide/Introduction

Introduction
****

This manual is incomplete: No attempt is made to explain basic concepts in a way suitable for novices. Thus, if you are interested, please get involved in improving this manual. The entire GNU community will benefit.

The GNU text utilities are mostly compatible with the POSIX.2 standard.

Please report bugs to 'bug-gnu-utils@prep.ai.mit.edu'. Remember to include the version number, machine architecture, input files, and any other information needed to reproduce the bug. See Bugs.

This manual is based on the Unix man pages in the distribution, which were originally written by David MacKenzie and updated by Jim Meyering. The original 'fmt' man page was written by Ross Paterson. Franc,ois Pinard did the initial conversion to Texinfo format. Karl Berry did the indexing, some reorganization, and editing of the results. Richard Stallman contributed his usual invaluable insights to the overall process.

1.3 textutils.guide/Common options

Common options
****

Certain options are available in all these programs. Rather than writing identical descriptions for each of the programs, they are described here. (In fact, every GNU program accepts (or should accept) these options.)

A few of these programs take arbitrary strings as arguments. In those cases, '--help' and '--version' are taken as these options only if there is one and exactly one command line argument.

'--help'
: Print a usage message listing all available options, then exit successfully.

'--version'
: Print the version number, then exit successfully.

1.4 textutils.guide/Output of entire files

```
                Output of entire files
**********************

   These commands read and write entire files, possibly transforming
them in some way.



                cat invocation
                              Concatenate and write files.

                tac invocation
                              Concatenate and write files in reverse.

                nl invocation
                               Number lines and write files.

                od invocation
                               Write files in octal or other formats.
```

1.5 textutils.guide/cat invocation

```
                'cat': Concatenate and write files
==================================

   'cat' copies each FILE ('-' means standard input), or standard input
if none are given, to standard output.  Synopsis:

     cat [OPTION] [FILE]...

   The program accepts the following options.  Also see See

                Common options
                .

'-A'
'--show-all'
     Equivalent to '-vET'.

'-b'
'--number-nonblank'
     Number all nonblank output lines, starting with 1.

'-e'
     Equivalent to '-vE'.

'-E'
'--show-ends'
     Display a '$' after the end of each line.
```

```
‘-n’
‘--number’
     Number all output lines, starting with 1.

‘-s’
‘--squeeze-blank’
     Replace multiple adjacent blank lines with a single blank line.

‘-t’
     Equivalent to ‘-vT’.

‘-T’
‘--show-tabs’
     Display TAB characters as ‘^I’.

‘-u’
     Ignored; for Unix compatibility.

‘-v’
‘--show-nonprinting’
     Display control characters except for LFD and TAB using ‘^’
     notation and precede characters that have the high bit set with
     ‘M-’.
```

1.6 textutils.guide/tac invocation

```
                ‘tac’: Concatenate and write files in reverse
============================================

   ‘tac’ copies each FILE (‘-’ means standard input), or standard input
if none are given, to standard output, reversing the records (lines by
default) in each separately.  Synopsis:

     tac [OPTION]... [FILE]...

   "Records" are separated by instances of a string (newline by
default)).  By default, this separator string is attached to the end of
the record that it follows in the file.

   The program accepts the following options.  Also see See

                Common options
                .

‘-b’
‘--before’
     The separator is attached to the beginning of the record that it
     precedes in the file.

‘-r’
‘--regex’
     Treat the separator string as a regular expression.
```

‘-s SEPARATOR’
‘--separator=SEPARATOR’
     Use SEPARATOR as the record separator, instead of newline.

1.7 textutils.guide/nl invocation

‘nl’: Number lines and write files
==========

‘nl’ writes each FILE (‘-’ means standard input), or standard input if none are given, to standard output, with line numbers added to some or all of the lines. Synopsis:

```
nl [OPTION]... [FILE]...
```

‘nl’ decomposes its input into (logical) pages; by default, the line number is reset to 1 at the top of each logical page. ‘nl’ treats all of the input files as a single document; it does not reset line numbers or logical pages between files.

A logical page consists of three sections: header, body, and footer. Any of the sections can be empty. Each can be numbered in a different style from the others.

The beginnings of the sections of logical pages are indicated in the input file by a line containing exactly one of these delimiter strings:

‘\:\:\:’
     start of header;

‘\:\:’
     start of body;

‘\:’
     start of footer.

The two characters from which these strings are made can be changed from ‘\’ and ‘:’ via options (see below), but the pattern and length of each string cannot be changed.

A section delimiter is replaced by an empty line on output. Any text that comes before the first section delimiter string in the input file is considered to be part of a body section, so ‘nl’ treats a file that contains no section delimiters as a single body section.

The program accepts the following options. Also see See

Common options
.

‘-b STYLE’
‘--body-numbering=STYLE’
     Select the numbering style for lines in the body section of each

```
     logical page.  When a line is not numbered, the current line number
     is not incremented, but the line number separator character is
     still prepended to the line.  The styles are:

    ‘a’
          number all lines,

    ‘t’
          number only nonempty lines (default for body),

    ‘n’
          do not number lines (default for header and footer),

    ‘pREGEXP’
          number only lines that contain a match for REGEXP.

‘-d CD’
‘--section-delimiter=CD’
     Set the section delimiter characters to CD; default is ‘\:’. If
     only C is given, the second remains ‘:’.  (Remember to protect ‘\’
     or other metacharacters from shell expansion with quotes or extra
     backslashes.)

‘-f STYLE’
‘--footer-numbering=STYLE’
     Analogous to ‘--body-numbering’.

‘-h STYLE’
‘--header-numbering=STYLE’
     Analogous to ‘--body-numbering’.

‘-i NUMBER’
‘--page-increment=NUMBER’
     Increment line numbers by NUMBER (default 1).

‘-l NUMBER’
‘--join-blank-lines=NUMBER’
     Consider NUMBER (default 1) consecutive empty lines to be one
     logical line for numbering, and only number the last one.  Where
     fewer than NUMBER consecutive empty lines occur, do not number
     them.  An empty line is one that contains no characters, not even
     spaces or tabs.

‘-n FORMAT’
‘--number-format=FORMAT’
     Select the line numbering format (default is ‘rn’):

    ‘ln’
          left justified, no leading zeros;

    ‘rn’
          right justified, no leading zeros;

    ‘rz’
          right justified, leading zeros.

‘-p’
```

‘--no-renumber’
     Do not reset the line number at the start of a logical page.

‘-s STRING’
‘--number-separator=STRING’
     Separate the line number from the text line in the output with
     STRING (default is TAB).

‘-v NUMBER’
‘--first-page=NUMBER’
     Set the initial line number on each logical page to NUMBER
     (default 1).

‘-w NUMBER’
‘--number-width=NUMBER’
     Use NUMBER characters for line numbers (default 6).

1.8 textutils.guide/od invocation

                ‘od’: Write files in octal or other formats
==

   ‘od’ writes an unambiguous representation of each FILE (‘-’ means
standard input), or standard input if none are given. Synopsis:

     od [OPTION]... [FILE]...
     od -C [FILE] [[+]OFFSET [[+]LABEL]]

   Each line of output consists of the offset in the input, followed by
groups of data from the file. By default, ‘od’ prints the offset in
octal, and each group of file data is two bytes of input printed as a
single octal number.

   The program accepts the following options. Also see See

                Common options
                .

‘-A RADIX’
‘--address-radix=RADIX’
     Select the base in which file offsets are printed. RADIX can be
     one of the following:

    ‘d’
          decimal;

    ‘o’
          octal;

    ‘x’
          hexadecimal;

    ‘n’

```
          none (do not print offsets).

     The default is octal.

‘-j BYTES’
‘--skip-bytes=BYTES’
     Skip BYTES input bytes before formatting and writing.  If BYTES
     begins with ‘0x’ or ‘0X’, it is interpreted in hexadecimal;
     otherwise, if it begins with ‘0’, in octal; otherwise, in decimal.
     Appending ‘b’ multiplies BYTES by 512, ‘k’ by 1024, and ‘m’ by
     1048576.

‘-N BYTES’
‘--read-bytes=BYTES’
     Output at most BYTES bytes of the input.  Prefixes and suffixes on
     ‘bytes’ are interpreted as for the ‘-j’ option.

‘-s [N]’
‘--strings[=N]’
     Instead of the normal output, output only "string constants": at
     least N (3 by default) consecutive ASCII graphic characters,
     followed by a null (zero) byte.

‘-t TYPE’
‘--format=TYPE’
     Select the format in which to output the file data.  TYPE is a
     string of one or more of the below type indicator characters.  If
     you include more than one type indicator character in a single TYPE
     string, or use this option more than once, ‘od’ writes one copy of
     each output line using each of the data types that you specified,
     in the order that you specified.

    ‘a’
          named character,

    ‘c’
          ASCII character or backslash escape,

    ‘d’
          signed decimal,

    ‘f’
          floating point,

    ‘o’
          octal,

    ‘u’
          unsigned decimal,

    ‘x’
          hexadecimal.

     The type ‘a’ outputs things like ‘sp’ for space, ‘nl’ for newline,
     and ‘nul’ for a null (zero) byte.  Type ‘c’ outputs ‘ ’, ‘\n’, and
     ‘\0’, respectively.
```

Except for types ‘a’ and ‘c’, you can specify the number of bytes to use in interpreting each number in the given data type by following the type indicator character with a decimal integer. Alternately, you can specify the size of one of the C compiler’s built-in data types by following the type indicator character with one of the following characters. For integers (‘d’, ‘o’, ‘u’, ‘x’):

‘C’
: char,

‘S’
: short,

‘I’
: int,

‘L’
: long.

For floating point (‘f’):

F
: float,

D
: double,

L
: long double.

‘-v’
‘--output-duplicates’
: Output consecutive lines that are identical. By default, when two or more consecutive output lines would be identical, ‘od’ outputs only the first line, and puts just an asterisk on the following line to indicate the elision.

‘-w [N]’
‘--width[=N]’
: Dump ‘n’ input bytes per output line. This must be a multiple of the least common multiple of the sizes associated with the specified output types. If N is omitted, the default is 32. If this option is not given at all, the default is 16.

The next several options map the old, pre-POSIX format specification options to the corresponding POSIX format specs. GNU ‘od’ accepts any combination of old- and new-style options. Format specification options accumulate.

‘-a’
: Output as named characters. Equivalent to ‘-ta’.

‘-b’
: Output as octal bytes. Equivalent to ‘-toC’.

‘-c’

```
     Output as ASCII characters or backslash escapes.  Equivalent to
     '-tc'.

'-d'
     Output as unsigned decimal shorts.  Equivalent to '-tu2'.

'-f'
     Output as floats.  Equivalent to '-tfF'.

'-h'
     Output as hexadecimal shorts.  Equivalent to '-tx2'.

'-i'
     Output as decimal shorts.  Equivalent to '-td2'.

'-l'
     Output as decimal longs.  Equivalent to '-td4'.

'-o'
     Output as octal shorts.  Equivalent to '-to2'.

'-x'
     Output as hexadecimal shorts.  Equivalent to '-tx2'.

'-C'
'--traditional'
     Recognize the pre-POSIX non-option arguments that traditional 'od'
     accepted.  The following syntax:

          od --traditional [FILE] [[+]OFFSET[.][b] [[+]LABEL[.][b]]]

     can be used to specify at most one file and optional arguments
     specifying an offset and a pseudo-start address, LABEL.  By
     default, OFFSET is interpreted as an octal number specifying how
     many input bytes to skip before formatting and writing.  The
     optional trailing decimal point forces the interpretation of
     OFFSET as a decimal number.  If no decimal is specified and the
     offset begins with '0x' or '0X' it is interpreted as a hexadecimal
     number.  If there is a trailing 'b', the number of bytes skipped
     will be OFFSET multiplied by 512.  The LABEL argument is
     interpreted just like OFFSET, but it specifies an initial
     pseudo-address.  The pseudo-addresses are displayed in parentheses
     following any normal address.
```

1.9 textutils.guide/Formatting file contents

```
                Formatting file contents
************************

   These commands reformat the contents of files.
```

```
                fmt invocation
                   Reformat paragraph text.

                pr invocation
                              Paginate or columnate files for printing.

                fold invocation
                            Wrap input lines to fit in specified width.
```

1.10 textutils.guide/fmt invocation

```
                ‘fmt’: Reformat paragraph text
==============================
```

‘fmt’ fills and joins lines to produce output lines of (at most) a given number of characters (75 by default). Synopsis:

```
     fmt [OPTION]... [FILE]...
```

‘fmt’ reads from the specified FILE arguments (or standard input if none), and writes to standard output.

By default, blank lines, spaces between words, and indentation are preserved in the output; successive input lines with different indentation are not joined; tabs are expanded on input and introduced on output.

‘fmt’ prefers breaking lines at the end of a sentence, and tries to avoid line breaks after the first word of a sentence or before the last word of a sentence. A "sentence break" is defined as either the end of a paragraph or a word ending in any of ‘.?!’, followed by two spaces or end of line, ignoring any intervening parentheses or quotes. Like TeX, ‘fmt’ reads entire "paragraphs" before choosing line breaks; the algorithm is a variant of that in "Breaking Paragraphs Into Lines" (Donald E. Knuth and Michael F. Plass, ‘Software--Practice and Experience’, 11 (1981), 1119-1184).

The program accepts the following options. Also see See

```
                Common options
                .
```

‘-c’
‘--crown-margin’
"Crown margin" mode: preserve the indentation of the first two lines within a paragraph, and align the left margin of each subsequent line with that of the second line.

‘-t’
‘--tagged-paragraph’
"Tagged paragraph" mode: like crown margin mode, except that if indentation of the first line of a paragraph is the same as the indentation of the second, the first line is treated as a one-line

```
     paragraph.

‘-s’
‘--split-only’
     Split lines only.  Do not join short lines to form longer ones.
     This prevents sample lines of code, and other such "formatted"
     text from being unduly combined.

‘-u’
‘--uniform-spacing’
     Uniform spacing.  Reduce spacing between words to one space, and
     spacing between sentences to two spaces.

‘-WIDTH’
‘-w WIDTH’
‘--width=WIDTH’
     Fill output lines up to WIDTH characters (default 75).  ‘fmt’
     initially tries to make lines about 7% shorter than this, to give
     it room to balance line lengths.

‘-p PREFIX’
‘--prefix=PREFIX’
     Only lines beginning with PREFIX (possibly preceded by whitespace)
     are subject to formatting. The prefix and any preceding whitespace
     is stripped for the formatting and then re-attached to each
     formatted output line.  One use is to format certain kinds of
     program comments, while leaving the code unchanged.
```

1.11 textutils.guide/pr invocation

```
                ‘pr’: Paginate or columnate files for printing
==============================================

   ‘pr’ writes each FILE (‘-’ means standard input), or standard input
if none are given, to standard output, paginating and optionally
outputting in multicolumn format.  Synopsis:

     pr [OPTION]... [FILE]...

   By default, a 5-line header is printed: two blank lines; a line with
the date, the filename, and the page count; and two more blank lines.  A
five line footer (entirely) is also printed.

   Form feeds in the input cause page breaks in the output.

   The program accepts the following options.  Also see See

               Common options
               .

‘+PAGE’
     Begin printing with page PAGE.
```

‘-COLUMN’
     Produce COLUMN-column output and print columns down. The column width is automatically decreased as COLUMN increases; unless you use the ‘-w’ option to increase the page width as well, this option might well cause some input to be truncated.

‘-a’
     Print columns across rather than down.

‘-b’
     Balance columns on the last page.

‘-c’
     Print control characters using hat notation (e.g., ‘^G’); print other unprintable characters in octal backslash notation. By default, unprintable characters are not changed.

‘-d’
     Double space the output.

‘-e[IN-TABCHAR[IN-TABWIDTH]]’
     Expand tabs to spaces on input. Optional argument IN-TABCHAR is the input tab character (default is TAB). Second optional argument IN-TABWIDTH is the input tab character’s width (default is 8).

‘-f’
‘-F’
     Use a formfeed instead of newlines to separate output pages.

‘-h HEADER’
     Replace the filename in the header with the string HEADER.

‘-i[OUT-TABCHAR[OUT-TABWIDTH]]’
     Replace spaces with tabs on output. Optional argument OUT-TABCHAR is the output tab character (default is TAB). Second optional argument OUT-TABWIDTH is the output tab character’s width (default is 8).

‘-l N’
     Set the page length to N (default 66) lines. If N is less than 10, the headers and footers are omitted, as if the ‘-t’ option had been given.

‘-m’
     Print all files in parallel, one in each column.

‘-n[NUMBER-SEPARATOR[DIGITS]]’
     Precede each column with a line number; with parallel files (‘-m’), precede each line with a line number. Optional argument NUMBER-SEPARATOR is the character to print after each number (default is TAB). Optional argument DIGITS is the number of digits per line number (default is 5).

‘-o N’
     Indent each line with N (default is zero) spaces wide, i.e., set the left margin. The total page width is ‘n’ plus the width set

```
     with the ‘-w’ option.

‘-r’
     Do not print a warning message when an argument FILE cannot be
     opened.  (The exit status will still be nonzero, however.)

‘-s[C]’
     Separate columns by the single character C.  If C is omitted, the
     default is space; if this option is omitted altogether, the
     default is TAB.

‘-t’
     Do not print the usual 5-line header and the 5-line footer on each
     page, and do not fill out the bottoms of pages (with blank lines or
     formfeeds).

‘-v’
     Print unprintable characters in octal backslash notation.

‘-w N’
     Set the page width to N (default is 72) columns.
```

1.12 textutils.guide/fold invocation

```
                ‘fold’: Wrap input lines to fit in specified width
==================================================

   ‘fold’ writes each FILE (‘-’ means standard input), or standard
input if none are given, to standard output, breaking long lines.
Synopsis:

     fold [OPTION]... [FILE]...

   By default, ‘fold’ breaks lines wider than 80 columns. The output is
split into as many lines as necessary.

   ‘fold’ counts screen columns by default; thus, a tab may count more
than one column, backspace decreases the column count, and carriage
return sets the column to zero.

   The program accepts the following options.  Also see See

               Common options
               .

‘-b’
‘--bytes’
     Count bytes rather than columns, so that tabs, backspaces, and
     carriage returns are each counted as taking up one column, just
     like other characters.

‘-s’
‘--spaces’
```

```
     Break at word boundaries: the line is broken after the last blank
     before the maximum line length.  If the line contains no such
     blanks, the line is broken at the maximum line length as usual.

‘-w WIDTH’
‘--width=WIDTH’
     Use a maximum line length of WIDTH columns instead of 80.
```

1.13 textutils.guide/Output of parts of files

```
                Output of parts of files
************************

   These commands output pieces of the input.


                head invocation
                            Output the first part of files.

                tail invocation
                            Output the last part of files.

                split invocation
                           Split a file into fixed-size pieces.

                csplit invocation
                          Split a file into context-determined pieces.
```

1.14 textutils.guide/head invocation

```
                ‘head’: Output the first part of files
======================================

   ‘head’ prints the first part (10 lines by default) of each FILE; it
reads from standard input if no files are given or when given a FILE of
‘-’.  Synopses:

     head [OPTION]... [FILE]...
     head -NUMBER [OPTION]... [FILE]...

   If more than one FILE is specicified, ‘head’ prints a one-line
header consisting of
     ==> FILENAME <==

before the output for each FILE.

   ‘head’ accepts two option formats: the new one, in which numbers are
```

```
arguments to the options (‘-q -n 1’), and the old one, in which the
number precedes any option letters (‘-1q’).

   The program accepts the following options.  Also see See

               Common options
               .

‘-COUNTOPTIONS’
     This option is only recognized if it is specified first.  COUNT is
     a decimal number optionally followed by a size letter (‘b’, ‘k’,
     ‘m’) as in ‘-c’, or ‘l’ to mean count by lines, or other option
     letters (‘cqv’).

‘-c BYTES’
‘--bytes=BYTES’
     Print the first BYTES bytes, instead of initial lines.  Appending
     ‘b’ multiplies BYTES by 512, ‘k’ by 1024, and ‘m’ by 1048576.

‘-n N’
‘--lines=N’
     Output the first N lines.

‘-q’
‘--quiet’
‘--silent’
     Never print filename headers.

‘-v’
‘--verbose’
     Always print filename headers.
```

1.15 textutils.guide/tail invocation

```
                ‘tail’: Output the last part of files
=====================================

   ‘tail’ prints the last part (10 lines by default) of each FILE; it
reads from standard input if no files are given or when given a FILE of
‘-’.  Synopses:

     tail [OPTION]... [FILE]...
     tail -NUMBER [OPTION]... [FILE]...
     tail +NUMBER [OPTION]... [FILE]...

   If more than one FILE is specified, ‘tail’ prints a one-line header
consisting of
     ==> FILENAME <==

before the output for each FILE.

   GNU ‘tail’ can output any amount of data (some other versions of
‘tail’ cannot).  It also has no ‘-r’ option (print in reverse), since
```

reversing a file is really a different job from printing the end of a file; BSD ‘tail’ (which is the one with ‘-r’) can only reverse files that are at most as large as its buffer, which is typically 32k. A more reliable and versatile way to reverse files is the GNU ‘tac’ command.

‘head’ accepts two option formats: the new one, in which numbers are arguments to the options (‘-n 1’), and the old one, in which the number precedes any option letters (‘-1’ or ‘+1’).

If any option-argument is a number N starting with a ‘+’, ‘tail’ begins printing with the Nth item from the start of each file, instead of from the end.

The program accepts the following options. Also see See

Common options
.

‘-COUNT’
‘+COUNT’
This option is only recognized if it is specified first. COUNT is a decimal number optionally followed by a size letter (‘b’, ‘k’, ‘m’) as in ‘-c’, or ‘l’ to mean count by lines, or other option letters (‘cfqv’).

‘-c BYTES’
‘--bytes=BYTES’
Output the last BYTES bytes, instead of final lines. Appending ‘b’ multiplies BYTES by 512, ‘k’ by 1024, and ‘m’ by 1048576.

‘-f’
‘--follow’
Loop forever trying to read more characters at the end of the file, presumably because the file is growing. Ignored if reading from a pipe. If more than one file is given, ‘tail’ prints a header whenever it gets output from a different file, to indicate which file that output is from.

‘-n N’
‘--lines=N’
Output the last N lines.

‘-q’
‘-quiet’
‘--silent’
Never print filename headers.

‘-v’
‘--verbose’
Always print filename headers.

1.16 textutils.guide/split invocation

```
                'split': Split a file into fixed-size pieces
============================================

   'split' creates output files containing consecutive sections of
INPUT (standard input if none is given or INPUT is '-').  Synopsis:

     split [OPTION] [INPUT [PREFIX]]

   By default, 'split' puts 1000 lines of INPUT (or whatever is left
over for the last section), into each output file.

   The output files' names consist of PREFIX ('x' by default) followed
by a group of letters 'aa', 'ab', and so on, such that concatenating
the output files in sorted order by filename produces the original
input file.  (If more than 676 output files are required, 'split' uses
'zaa', 'zab', etc.)

   The program accepts the following options.  Also see See

               Common options
               .

'-LINES'
'-l LINES'
'--lines=LINES'
     Put LINES lines of INPUT into each output file.

'-b BYTES'
'--bytes=BYTES'
     Put the first BYTES bytes of INPUT into each output file.
     Appending 'b' multiplies BYTES by 512, 'k' by 1024, and 'm' by
     1048576.

'-C BYTES'
'--line-bytes=BYTES'
     Put into each output file as many complete lines of INPUT as
     possible without exceeding BYTES bytes.  For lines longer than
     BYTES bytes, put BYTES bytes into each output file until less than
     BYTES bytes of the line are left, then continue normally.  BYTES
     has the same format as for the '--bytes' option.
```

1.17 textutils.guide/csplit invocation

```
                'csplit': Split a file into context-determined pieces
=====================================================

   'csplit' creates zero or more output files containing sections of
INPUT (standard input if INPUT is '-').  Synopsis:

     csplit [OPTION]... INPUT PATTERN...
```

The contents of the output files are determined by the PATTERN arguments, as detailed below. An error occurs if a PATTERN argument refers to a nonexistent line of the input file (e.g., if no remaining line matches a given regular expression). After every PATTERN has been matched, any remaining input is copied into one last output file.

By default, 'csplit' prints the number of bytes written to each output file after it has been created.

The types of pattern arguments are:

'N'
: Create an output file containing the input up to but not including line N (a positive integer). If followed by a repeat count, also create an output file containing the next LINE lines of the input file once for each repeat.

'/REGEXP/[OFFSET]'
: Create an output file containing the current line up to (but not including) the next line of the input file that contains a match for REGEXP. The optional OFFSET is a '+' or '-' followed by a positive integer. If it is given, the input up to the matching line plus or minus OFFSET is put into the output file, and the line after that begins the next section of input.

'%REGEXP%[OFFSET]'
: Like the previous type, except that it does not create an output file, so that section of the input file is effectively ignored.

'{REPEAT-COUNT}'
: Repeat the previous pattern REPEAT-COUNT additional times. REPEAT-COUNT can either be a positive integer or an asterisk, meaning repeat as many times as necessary until the input is exausted.

The output files' names consist of a prefix ('xx' by default) followed by a suffix. By default, the suffix is an ascending sequence of two-digit decimal numbers from '00' and up to '99'. In any case, concatenating the output files in sorted order by file name produces the original input file.

By default, if 'csplit' encounters an error or receives a hangup, interrupt, quit, or terminate signal, it removes any output files that it has created so far before it exits.

The program accepts the following options. Also see See

Common options
.

'-f PREFIX'
'--prefix=PREFIX'
: Use PREFIX as the output filename prefix.

'-b SUFFIX'
'--suffix=SUFFIX'

```
     Use SUFFIX as the output filename suffix.  When this option is
     specified, the suffix string must include exactly one
     'printf(3)'-style conversion specification, possibly including
     format specification flags, a field width, a precision
     specifications, or all of these kinds of modifiers.  The format
     letter must convert a binary integer argument to readable form;
     thus, only 'd', 'i', 'u', 'o', 'x', and 'X' conversions are
     allowed.  The entire SUFFIX is given (with the current output file
     number) to 'sprintf(3)' to form the filename suffixes for each of
     the individual output files in turn.  If this option is used, the
     '--digits' option is ignored.

'-n DIGITS'
'--digits=DIGITS'
     Use output filenames containing numbers that are DIGITS digits
     long instead of the default 2.

'-k'
'--keep-files'
     Do not remove output files when errors are encountered.

'-z'
'--elide-empty-files'
     Suppress the generation of zero-length output files.  (In cases
     where the section delimiters of the input file are supposed to
     mark the first lines of each of the sections, the first output
     file will generally be a zero-length file unless you use this
     option.)  The output file sequence numbers always run
     consecutively starting from 0, even when this option is specified.

'-s'
'-q'
'--silent'
'--quiet'
     Do not print counts of output file sizes.
```

1.18 textutils.guide/Summarizing files

```
                Summarizing files
*****************

   These commands generate just a few numbers representing entire
contents of files.



                wc invocation
                               Print byte, word, and line counts.

                sum invocation
                             Print checksum and block counts.

                cksum invocation
```

Print CRC checksum and byte counts.

1.19 textutils.guide/wc invocation

'wc': Print byte, word, and line counts
==========

'wc' counts the number of bytes, whitespace-separated words, and newlines in each given FILE, or standard input if none are given or for a FILE of '-'. Synopsis:

```
wc [OPTION]... [FILE]...
```

'wc' prints one line of counts for each file, and if the file was given as an argument, it prints the filename following the counts. If more than one FILE is given, 'wc' prints a final line containing the cumulative counts, with the filename 'total'. The counts are printed in this order: lines, words, bytes.

By default, 'wc' prints all three counts. Options can specify that only certain counts be printed. Options do not undo others previously given, so

```
wc --bytes --words
```

prints both the byte counts and the word counts.

The program accepts the following options. Also see See

Common options
.

'-c'
'--bytes'
'--chars'
Print only the byte counts.

'-w'
'--words'
Print only the word counts.

'-l'
'--lines'
Print only the newline counts.

1.20 textutils.guide/sum invocation

'sum': Print checksum and block counts
==========

'sum' computes a 16-bit checksum for each given FILE, or standard input if none are given or for a FILE of '-'. Synopsis:

```
sum [OPTION]... [FILE]...
```

'sum' prints the checksum for each FILE followed by the number of blocks in the file (rounded up). If more than one FILE is given, filenames are also printed (by default). (With the '--sysv' option, corresponding file name are printed when there is at least one file argument.)

By default, GNU 'sum' computes checksums using an algorithm compatible with BSD 'sum' and prints file sizes in units of 1024-byte blocks.

The program accepts the following options. Also see See

Common options
.

'-r'
Use the default (BSD compatible) algorithm. This option is included for compatibility with the System V 'sum'. Unless '-s' was also given, it has no effect.

'-s'
'--sysv'
Compute checksums using an algorithm compatible with System V 'sum''s default, and print file sizes in units of 512-byte blocks.

'sum' is provided for compatibility; the 'cksum' program (see next section) is preferable in new applications.

1.21 textutils.guide/cksum invocation

'cksum': Print CRC checksum and byte counts
==========

'cksum' computes a cyclic redundancy check (CRC) checksum for each given FILE, or standard input if none are given or for a FILE of '-'. Synopsis:

Synopsis:

```
cksum [OPTION]... [FILE]...
```

'cksum' prints the CRC for each file along with the number of bytes in the file, and the filename unless no arguments were given.

‘cksum’ is typically used to ensure that files have been transferred by unreliable means (e.g., netnews) have not been corrupted, by comparing the ‘cksum’ output for the received files with the ‘cksum’ output for the original files (usually given in the distribution).

The CRC algorithm is specified by the POSIX.2 standard. It is not compatible with the BSD or System V ‘sum’ programs; it is more robust.

1.22 textutils.guide/Operating on sorted files

```
                Operating on sorted files
************************
```

These commands work with (or produce) sorted files.

sort invocation
: Sort text files.

uniq invocation
: Uniqify files.

comm invocation
: Compare two sorted files line by line.

1.23 textutils.guide/sort invocation

```
                ‘sort’: Sort text files
=======================
```

‘sort’ sorts, merges, or compares all the lines from the given files, or standard input if none are given or for a FILE of ‘-’. By default, ‘sort’ writes the results to standard output. Synopsis:

```
sort [OPTION]... [FILE]...
```

‘sort’ has three modes of operation: sort (the default), merge, and check for sortedness. The following options change the operation mode:

‘-c’
: Check whether the given files are already sorted: if they are not all sorted, print an error message and exit with a status of 1.

‘-m’
: Merge the given files by sorting them as a group. Each input file must always be individually sorted. It always works to sort instead of merge; merging is provided because it is faster, in the case where it works.

A pair of lines is compared as follows: if any key fields have been specified, ‘sort’ compares each pair of fields, in the order specified on the command line, according to the associated ordering options, until a difference is found or no fields are left.

If any of the global options ‘Mbdfinr’ are given but no key fields are specified, ‘sort’ compares the entire lines according to the global options.

Finally, as a last resort when all keys compare equal (or if no ordering options were specified at all), ‘sort’ compares the lines byte by byte in machine collating sequence. The last resort comparison honors the ‘-r’ global option. The ‘-s’ (stable) option disables this last-resort comparison so that lines in which all fields compare equal are left in their original relative order. If no fields or global options are specified, ‘-s’ has no effect.

GNU ‘sort’ (as specified for all GNU utilities) has no limits on input line length or restrictions on bytes allowed within lines. In addition, if the final byte of an input file is not a newline, GNU ‘sort’ silently supplies one.

If the environment variable ‘TMPDIR’ is set, ‘sort’ uses its value as the directory for temporary files instead of ‘/tmp’. The ‘-T TEMPDIR’ option in turn overrides the environment variable.

The following options affect the ordering of output lines. They may be specified globally or as part of a specific key field. If no key fields are specified, global options apply to comparison of entire lines; otherwise the global options are inherited by key fields that do not specify any special options of their own.

‘-b’
: Ignore leading blanks when finding sort keys in each line.

‘-d’
: Sort in "phone directory" order: ignore all characters except letters, digits and blanks when sorting.

‘-f’
: Fold lowercase characters into the equivalent uppercase characters when sorting so that, for example, ‘b’ and ‘B’ sort as equal.

‘-i’
: Ignore characters outside the printable ASCII range 040-0176 octal (inclusive) when sorting.

‘-M’
: An initial string, consisting of any amount of whitespace, followed by three letters abbreviating a month name, is folded to UPPER case and compared in the order ‘JAN’ < ‘FEB’ < ... < ‘DEC’. Invalid names compare low to valid names.

‘-n’
: Sort numerically: the number begins each line; specifically, it consists of optional whitespace, an optional ‘-’ sign, and zero or

more digits, optionally followed by a decimal point and zero or more digits.

‘-r’
Reverse the result of comparison, so that lines with greater key values appear earlier in the output instead of later.

Other options are:

‘-o OUTPUT-FILE’
Write output to OUTPUT-FILE instead of standard output. If OUTPUT-FILE is one of the input files, ‘sort’ copies it to a temporary file before sorting and writing the output to OUTPUT-FILE.

‘-t SEPARATOR’
Use character SEPARATOR as the field separator when finding the sort keys in each line. By default, fields are separated by the empty string between a non-whitespace character and a whitespace character. That is, given the input line ‘ foo bar’, ‘sort’ breaks it into fields ‘ foo’ and ‘ bar’. The field separator is not considered to be part of either the field preceding or the field following.

‘-u’
For the default case or the ‘-m’ option, only output the first of a sequence of lines that compare equal. For the ‘-c’ option, check that no pair of consecutive lines compares equal.

‘+POS1[-POS2]’
Specify a field within each line to use as a sorting key. The field consists of the portion of the line starting at POS1 and up to (but not including) POS2 (or to the end of the line if POS2 is not given). The fields and character positions are numbered starting with 0.

‘-k POS1[,POS2]’
An alternate syntax for specifying sorting keys. The fields and character positions are numbered starting with 1.

A position has the form ‘F.C’, where F is the number of the field to use and C is the number of the first character from the beginning of the field (for ‘+POS’) or from the end of the previous field (for ‘-POS’). The ‘.C’ part of a position may be omitted in which case it is taken to be the first character in the field. If the ‘-b’ option has been given, the ‘.C’ part of a field specification is counted from the first nonblank character of the field (for ‘+POS’) or from the first nonblank character following the previous field (for ‘-POS’).

A ‘+POS’ or ‘-POS’ argument may also have any of the option letters ‘Mbdfinr’ appended to it, in which case the global ordering options are not used for that particular field. The ‘-b’ option may be independently attached to either or both of the ‘+POS’ and ‘-POS’ parts of a field specification, and if it is inherited from the global options it will be attached to both. If a ‘-n’ or ‘-M’ option is used, thus implying a ‘-b’ option, the ‘-b’ option is taken to apply to both the ‘+POS’ and the ‘-POS’ parts of a key specification. Keys may span

multiple fields.

   In addition, when GNU ‘sort’ is invoked with exactly one argument, options ‘--help’ and ‘--version’ are recognized. See
               Common options
               .

   Historical (BSD and System V) implementations of ‘sort’ have differed in their interpretation of some options, particularly ‘-b’, ‘-f’, and ‘-n’. GNU sort follows the POSIX behavior, which is usually (but not always!) like the System V behavior. According to POSIX, ‘-n’ no longer implies ‘-b’. For consistency, ‘-M’ has been changed in the same way. This may affect the meaning of character positions in field specifications in obscure cases. The only fix is to add an explicit ‘-b’.

1.24 textutils.guide/uniq invocation

               ‘uniq’: Uniqify files
====================

   ‘uniq’ writes the unique lines in the given ‘input’, or standard input if nothing is given or for an INPUT name of ‘-’. Synopsis:

     uniq [OPTION]... [INPUT [OUTPUT]]

   By default, ‘uniq’ prints the unique lines in a sorted file, i.e., discards all but one of identical successive lines. Optionally, it can instead show only lines that appear exactly once, or lines that appear more than once.

   The input must be sorted. If your input is not sorted, perhaps you want to use ‘sort -u’.

   If no OUTPUT file is specified, ‘uniq’ writes to standard output.

   The program accepts the following options. Also see See

               Common options
               .

‘-N’
‘-f N’
‘--skip-fields=N’
     Skip N fields on each line before checking for uniqueness. Fields are sequences of non-space non-tab characters that are separated from each other by at least one spaces or tabs.

‘+N’
‘-s N’
‘--skip-chars=N’
     Skip N characters before checking for uniqueness. If you use both the field and character skipping options, fields are skipped over

```
     first.

‘-c’
‘--count’
     Print the number of times each line occurred along with the line.

‘-d’
‘--repeated’
     Print only duplicate lines.

‘-u’
‘--unique’
     Print only unique lines.

‘-w N’
‘--check-chars=N’
     Compare N characters on each line (after skipping any specified
     fields and characters).  By default the entire rest of the lines
     are compared.
```

1.25 textutils.guide/comm invocation

```
                ‘comm’: Compare two sorted files line by line
==============================================

   ‘comm’ writes to standard output lines that are common, and lines
that are unique, to two input files; a filename of ‘-’ means standard
input.  Synopsis:

     comm [OPTION]... FILE1 FILE2

   The input files must be sorted before ‘comm’ can be used.

   With no options, ‘comm’ produces three column output.  Column one
contains lines unique to FILE1, column two contains lines unique to
FILE2, and column three contains lines common to both files.

   The options ‘-1’, ‘-2’, and ‘-3’ suppress printing of the
corresponding columns.  Also see See
                Common options
                .
```

1.26 textutils.guide/Operating on fields within a line

```
                Operating on fields within a line
*********************************
```

```
cut invocation
              Print selected parts of lines.

paste invocation
            Merge lines of files.

join invocation
             Join lines on a common field.
```

1.27 textutils.guide/cut invocation

```
                'cut': Print selected parts of lines
====================================
```

'cut' writes to standard output selected parts of each line of each input file, or standard input if no files are given or for a filename of '-'. Synopsis:

```
     cut [OPTION]... [FILE]...
```

In the table which follows, the BYTE-LIST, CHARACTER-LIST, and FIELD-LIST are one or more numbers or ranges (two numbers separated by a dash) separated by commas. Bytes, characters, and fields are numbered from starting at 1. Incomplete ranges may be given: '-M' means '1-M'; 'N-' means 'N' through end of line or last field.

The program accepts the following options. Also see See

```
               Common options
               .

'-b BYTE-LIST'
'--bytes=BYTE-LIST'
     Print only the bytes in positions listed in BYTE-LIST. Tabs and
     backspaces are treated like any other character; they take up 1
     byte.

'-c CHARACTER-LIST'
'--characters=CHARACTER-LIST'
     Print only characters in positions listed in CHARACTER-LIST. The
     same as '-b' for now, but internationalization will change that.
     Tabs and backspaces are treated like any other character; they
     take up 1 character.

'-f FIELD-LIST'
'--fields=FIELD-LIST'
     Print only the fields listed in FIELD-LIST. Fields are separated
     by a TAB by default.

'-d DELIM'
'--delimiter=DELIM'
     For '-f', fields are separated by the first character in DELIM
```

```
     (default is TAB).

'-n'
     Do not split multibyte characters (no-op for now).

'-s'
'--only-delimited'
     For '-f', do not print lines that do not contain the field
     separator character.
```

1.28 textutils.guide/paste invocation

```
                'paste': Merge lines of files
============================

   'paste' writes to standard output lines consisting of sequentially
corresponding lines of each given file, separated by TAB.  Standard
input is used for a filename of '-' or if no input files are given.

   Synopsis:

     paste [OPTION]... [FILE]...

   The program accepts the following options.  Also see See

                Common options
                .

'-s'
'--serial'
     Paste the lines of one file at a time rather than one line from
     each file.

'-d DELIM-LIST'
'--delimiters DELIM-LIST'
     Consecutively use the characters in DELIM-LIST instead of TAB to
     separate merged lines.  When DELIM-LIST is exhausted, start again
     at its beginning.
```

1.29 textutils.guide/join invocation

```
                'join': Join lines on a common field
====================================

   'join' writes to standard output a line for each pair of input lines
that have identical join fields.  Synopsis:

     join [OPTION]... FILE1 FILE2
```

Either FILE1 or FILE2 (but not both) can be '-', meaning standard input. FILE1 and FILE2 should be already sorted in increasing order (not numerically) on the join fields; unless the '-t' option is given, they should be sorted ignoring blanks at the start of the line, as in 'sort -b'.

The defaults are: the join field is the first field in each line; fields in the input are separated by one or more blanks, with leading blanks on the line ignored; fields in the output are separated by a space; each output line consists of the join field, the remaining fields from FILE1, then the remaining fields from FILE2.

The program accepts the following options. Also see See

Common options
.

'-a FILE-NUMBER'
Print a line for each unpairable line in file FILE-NUMBER (either '1' or '2'), in addition to the normal output.

'-e STRING'
Replace those output fields that are missing in the input with STRING.

'-1 FIELD'
'-j1 FIELD'
Join on field FIELD (a positive integer) of file 1.

'-2 FIELD'
'-j2 FIELD'
Join on field FIELD (a positive integer) of file 2.

'-j FIELD'
Equivalent to '-1 FIELD -2 FIELD'.

'-o FIELD-LIST...'
Construct each output line according to the format in FIELD-LIST. Each element in FIELD-LIST consists of a file number (either 1 or 2), a period, and a field number (a positive integer). The elements in the list are separated by commas or blanks. Multiple FIELD-LIST arguments can be given after a single '-o' option; the values of all lists given with '-o' are concatenated together.

'-t CHAR'
Use character CHAR as the input and output field separator.

'-v FILE-NUMBER'
Print a line for each unpairable line in file FILE-NUMBER (either 1 or 2), instead of the normal output.

In addition, when GNU 'join' is invoked with exactly one argument, options '--help' and '--version' are recognized. See
Common options
.

1.30 textutils.guide/Operating on characters

```
                Operating on characters
**********************

   This commands operate on individual characters.



                tr invocation
                               Translate, squeeze, and/or delete characters.

                expand invocation
                           Convert tabs to spaces.

                unexpand invocation
                         Convert spaces to tabs.
```

1.31 textutils.guide/tr invocation

```
                'tr': Translate, squeeze, and/or delete characters
==================================================

   Synopsis:

     tr [OPTION]... SET1 [SET2]

   'tr' copies standard input to standard output, performing one of the
following operations:

   * translate, and optionally squeeze repeated characters in the
     result,

   * squeeze repeated characters,

   * delete characters,

   * delete characters, then squeeze repeated characters from the
     result.

   The SET1 and (if given) SET2 arguments define ordered sets of
characters, referred to below as SET1 and SET2.  These sets are the
characters of the input that 'tr' operates on.  The '--complement'
('-c') option replaces SET1 with its complement (all of the characters
that are not in SET1).
```

1.32 textutils.guide/Character sets

```
Specifying sets of characters
-----------------------------
```

The format of the SET1 and SET2 arguments resembles the format of regular expressions; however, they are not regular expressions, only lists of characters. Most characters simply represent themselves in these strings, but the strings can contain the shorthands listed below, for convenience. Some of them can be used only in SET1 or SET2, as noted below.

Backslash escapes.
A backslash followed by a character not listed below causes an error message.

‘\a’
Control-G,

‘\b’
Control-H,

‘\f’
Control-L,

‘\n’
Control-J,

‘\r’
Control-M,

‘\t’
Control-I,

‘\v’
Control-K,

‘\OOO’
The character with the value given by OOO, which is 1 to 3 octal digits,

‘\’
A backslash.

Ranges.
The notation ‘M-N’ expands to all of the characters from M through N, in ascending order. M should collate before N; if it doesn’t, an error results. As an example, ‘0-9’ is the same as ‘0123456789’. Although GNU ‘tr’ does not support the System V syntax that uses square brackets to enclose ranges, translations specified in that format will still work as long as the brackets in STRING1 correspond to identical brackets in STRING2.

Repeated characters.
The notation ‘[C*N]’ in SET2 expands to N copies of character C. Thus, ‘[y*6]’ is the same as ‘yyyyyy’. The notation ‘[C*]’ in STRING2 expands to as many copies of C as are needed to make SET2 as long as SET1. If N begins with ‘0’, it is interpreted in octal, otherwise in decimal.

Character classes.
The notation ‘[:CLASS:]’ expands to all of the characters in the (predefined) class CLASS. The characters expand in no particular order, except for the ‘upper’ and ‘lower’ classes, which expand in ascending order. When the ‘--delete’ (‘-d’) and ‘--squeeze-repeats’ (‘-s’) options are both given, any character class can be used in SET2. Otherwise, only the character classes ‘lower’ and ‘upper’ are accepted in SET2, and then only if the corresponding character class (‘upper’ and ‘lower’, respectively) is specified in the same relative position in SET1. Doing this specifies case conversion. The class names are given below; an error results when an invalid class name is given.

‘alnum’
Letters and digits.

‘alpha’
Letters.

‘blank’
Horizontal whitespace.

‘cntrl’
Control characters.

‘digit’
Digits.

‘graph’
Printable characters, not including space.

‘lower’
Lowercase letters.

‘print’
Printable characters, including space.

‘punct’

```
          Punctuation characters.

     'space'
          Horizontal or vertical whitespace.

     'upper'
          Uppercase letters.

     'xdigit'
          Hexadecimal digits.

Equivalence classes.
     The syntax '[=C=]' expands to all of the characters that are
     equivalent to C, in no particular order.  Equivalence classes are
     a relatively recent invention intended to support non-English
     alphabets.  But there seems to be no standard way to define them
     or determine their contents.  Therefore, they are not fully
     implemented in GNU 'tr'; each character's equivalence class
     consists only of that character, which is of no particular use.
```

1.33 textutils.guide/Translating

```
Translating
-----------

   'tr' performs translation when SET1 and SET2 are both given and the
'--delete' ('-d') option is not given.  'tr' translates each character
of its input that is in SET1 to the corresponding character in SET2.
Characters not in SET1 are passed through unchanged.  When a character
appears more than once in SET1 and the corresponding characters in SET2
are not all the same, only the final one is used.  For example, these
two commands are equivalent:

     tr aaa xyz
     tr a z

   A common use of 'tr' is to convert lowercase characters to
uppercase.  This can be done in many ways.  Here are three of them:

     tr abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ
     tr a-z A-Z
     tr '[:lower:]' '[:upper:]'

   When 'tr' is performing translation, SET1 and SET2 typically have
the same length.  If SET1 is shorter than SET2, the extra characters at
the end of SET2 are ignored.

   On the other hand, making SET1 longer than SET2 is not portable;
POSIX.2 says that the result is undefined.  In this situation, BSD 'tr'
pads SET2 to the length of SET1 by repeating the last character of SET2
as many times as necessary.  System V 'tr' truncates SET1 to the length
of SET2.

   By default, GNU 'tr' handles this case like BSD 'tr'.  When the
```

‘--truncate-set1’ (‘-t’) option is given, GNU ‘tr’ handles this case like the System V ‘tr’ instead. This option is ignored for operations other than translation.

Acting like System V ‘tr’ in this case breaks the relatively common BSD idiom:

```
tr -cs A-Za-z0-9 '\012'
```

because it converts only zero bytes (the first element in the complement of SET1), rather than all non-alphanumerics, to newlines.

1.34 textutils.guide/Squeezing

Squeezing repeats and deleting

When given just the ‘--delete’ (‘-d’) option, ‘tr’ removes any input characters that are in SET1.

When given just the ‘--squeeze-repeats’ (‘-s’) option, ‘tr’ replaces each input sequence of a repeated character that is in SET1 with a single occurrence of that character.

When given both ‘--delete’ and ‘--squeeze-repeats’, ‘tr’ first performs any deletions using SET1, then squeezes repeats from any remaining characters using SET2.

The ‘--squeeze-repeats’ option may also be used when translating, in which case ‘tr’ first performs translation, then squeezes repeats from any remaining characters using SET2.

Here are some examples to illustrate various combinations of options:

* Remove all zero bytes:

```
tr -d '\000'
```

* Put all words on lines by themselves. This converts all non-alphanumeric characters to newlines, then squeezes each string of repeated newlines into a single newline:

```
tr -cs '[a-zA-Z0-9]' '[\n*]'
```

* Convert each sequence of repeated newlines to a single newline:

```
tr -s '\n'
```

1.35 textutils.guide/Warnings in tr

```
Warning messages
----------------
```

   Setting the environment variable ‘POSIXLY_CORRECT’ turns off the following warning and error messages, for strict compliance with POSIX.2. Otherwise, the following diagnostics are issued:

1. When the ‘--delete’ option is given but ‘--squeeze-repeats’ is not, and SET2 is given, GNU ‘tr’ by default prints a usage message and exits, because SET2 would not be used. The POSIX specification says that SET2 must be ignored in this case. Silently ignoring arguments is a bad idea.

2. When an ambiguous octal escape is given. For example, ‘\400’ is actually ‘\40’ followed by the digit ‘0’, because the value 400 octal does not fit into a single byte.

   GNU ‘tr’ does not provide complete BSD or System V compatibility. For example, it is impossible to disable interpretation of the POSIX constructs ‘[:alpha:]’, ‘[=c=]’, and ‘[c*10]’. Also, GNU ‘tr’ does not delete zero bytes automatically, unlike traditional Unix versions, which provide no way to preserve zero bytes.

1.36 textutils.guide/expand invocation

```
                ‘expand’: Convert tabs to spaces
================================
```

   ‘expand’ writes the contents of each given FILE, or standard input if none are given or for a FILE of ‘-’, to standard output, with tab characters converted to the appropriate number of spaces. Synopsis:

```
     expand [OPTION]... [FILE]...
```

   By default, ‘expand’ converts all tabs to spaces. It preserves backspace characters in the output; they decrement the column count for tab calculations. The default action is equivalent to ‘-8’ (set tabs every 8 columns).

   The program accepts the following options. Also see See

```
               Common options
               .

‘-TAB1[,TAB2]...’
‘-t TAB1[,TAB2]...’
‘--tabs=TAB1[,TAB2]...’
     If only one tab stop is given, set the tabs TAB1 spaces apart
     (default is 8). Otherwise, set the tabs at columns TAB1, TAB2,
     ... (numbered from 0), and replace any tabs beyond the last
     tabstop given with single spaces. If the tabstops are specified
     with the ‘-t’ or ‘--tabs’ option, they can be separated by blanks
```

```
     as well as by commas.

‘-i’
‘--initial’
     Only convert initial tabs (those that precede all non-space or
     non-tab characters) on each line to spaces.
```

1.37 textutils.guide/unexpand invocation

```
                ‘unexpand’: Convert spaces to tabs
==================================

   ‘unexpand’ writes the contents of each given FILE, or standard input
if none are given or for a FILE of ‘-’, to standard output, with
strings of two or more space or tab characters converted to as many
tabs as possible followed by as many spaces as are needed.  Synopsis:

     unexpand [OPTION]... [FILE]...

   By default, ‘unexpand’ converts only initial spaces and tabs (those
that precede all non space or tab characters) on each line.  It
preserves backspace characters in the output; they decrement the column
count for tab calculations.  By default, tabs are set at every 8th
column.

   The program accepts the following options.  Also see See

               Common options
               .

‘-TAB1[,TAB2]...’
‘-t TAB1[,TAB2]...’
‘--tabs=TAB1[,TAB2]...’
     If only one tab stop is given, set the tabs TAB1 spaces apart
     instead of the default 8.  Otherwise, set the tabs at columns
     TAB1, TAB2, ... (numbered from 0), and leave spaces and tabs
     beyond the tabstops given unchanged.  If the tabstops are specified
     with the ‘-t’ or ‘--tabs’ option, they can be separated by blanks
     as well as by commas.  This option implies the ‘-a’ option.

‘-a’
‘--all’
     Convert all strings of two or more spaces or tabs, not just initial
     ones, to tabs.
```

1.38 textutils.guide/Index

Index

*****

```
-TAB
                                expand invocation
-TAB
                                unexpand invocation
-WIDTH
                              fmt invocation
-a
                                  unexpand invocation
-A
                                  cat invocation
-a
                                  join invocation
-a
                                  od invocation
-A
                                  od invocation
-a
                                  pr invocation
-b
                                  cut invocation
-b
                                  tac invocation
-b
                                  csplit invocation
-b
                                  pr invocation
-b
                                  split invocation
-b
                                  nl invocation
-b
                                  cat invocation
-b
                                  sort invocation
-b
                                  od invocation
-b
                                  fold invocation
```

-c
tail invocation

-c
wc invocation

-c
fmt invocation

-c
sort invocation

-C
split invocation

-c
head invocation

-c
od invocation

-c
pr invocation

-c
cut invocation

-c
uniq invocation

-d
uniq invocation

-d
cut invocation

-d
paste invocation

-d
od invocation

-d
pr invocation

-d
sort invocation

-d
nl invocation

-E
cat invocation

-e
cat invocation

`-e`
: join invocation

`-e`
: pr invocation

`-f`
: tail invocation

`-f`
: cut invocation

`-f`
: sort invocation

`-f`
: uniq invocation

`-f`
: csplit invocation

`-F`
: pr invocation

`-f`
: nl invocation

`-f`
: od invocation

`-f`
: pr invocation

`-h`
: od invocation

`-h`
: nl invocation

`-h`
: pr invocation

`-i`
: od invocation

`-i`
: nl invocation

`-i`
: sort invocation

`-i`
: pr invocation

`-i`
: expand invocation

```
-j
                                od invocation
-j1
                               join invocation
-j2
                               join invocation
-k
                                csplit invocation
-l
                                od invocation
-l
                                nl invocation
-l
                                pr invocation
-l
                                wc invocation
-l
                                split invocation
-M
                                sort invocation
-m
                                sort invocation
-m
                                pr invocation
-n
                                pr invocation
-n
                                cat invocation
-n
                                sort invocation
-n
                                csplit invocation
-n
                                cut invocation
-n
                                nl invocation
-N
                                od invocation
```

-n	head invocation
-n	tail invocation
-o	od invocation
-o	sort invocation
-o	pr invocation
-p	nl invocation
-q	csplit invocation
-q	tail invocation
-q	head invocation
-r	sum invocation
-r	sort invocation
-r	pr invocation
-r	tac invocation
-s	csplit invocation
-s	uniq invocation
-s	sum invocation
-s	tac invocation
-s	od invocation
-s	nl invocation

```
-s
                                pr invocation
-s
                                cut invocation
-s
                                fold invocation
-s
                                fmt invocation
-s
                                paste invocation
-s
                                cat invocation
-t
                                unexpand invocation
-t
                                od invocation
-t
                                expand invocation
-t
                                cat invocation
-t
                                fmt invocation
-t
                                pr invocation
-t
                                sort invocation
-T
                                cat invocation
-u
                                cat invocation
-u
                                sort invocation
-u
                                uniq invocation
-u
                                fmt invocation
-v
                                tail invocation
```