diff

COLLABORATORS

	TITLE : diff		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		December 7, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

diff

1.1 diff.guide

This file documents the the GNU 'diff', 'diff3', 'sdiff', and 'cmp'
commands for showing the differences between text files and the 'patch'
command for using their output to update files.

This is Edition 1.2, for 'diff' 2.4 and 'patch' 2.1.

1.2 diff.guide/Overview

Overview
****

   Computer users often find occasion to ask how two files differ.
Perhaps one file is a newer version of the other file. Or maybe the
two files started out as identical copies but were changed by different
people.

   You can use the 'diff' command to show differences between two
files, or each corresponding file in two directories. 'diff' outputs
differences between files line by line in any of several formats,
selectable by command line options. This set of differences is often
called a "diff" or "patch". For files that are identical, 'diff'
normally produces no output; for binary (non-text) files, 'diff'
normally reports only that they are different.

   You can use the 'cmp' command to show the offsets and line numbers
where two files differ. 'cmp' can also show all the characters that
differ between the two files, side by side. Another way to compare two
files character by character is the Emacs command 'M-x
compare-windows'. See Other Window, for more information on that
command.

   You can use the 'diff3' command to show differences among three
files. When two people have made independent changes to a common
original, 'diff3' can report the differences between the original and
the two changed versions, and can produce a merged file that contains
both persons' changes together with warnings about conflicts.

   You can use the 'sdiff' command to merge two files interactively.

   You can use the set of differences produced by 'diff' to distribute
updates to text files (such as program source code) to other people.
This method is especially useful when the differences are small compared
to the complete files. Given 'diff' output, you can use the 'patch'
program to update, or "patch", a copy of the file. If you think of
'diff' as subtracting one file from another to produce their
difference, you can think of 'patch' as adding the difference to one
file to reproduce the other.

   This manual first concentrates on making diffs, and later shows how
to use diffs to update files.

   GNU 'diff' was written by Mike Haertel, David Hayes, Richard
Stallman, Len Tower, and Paul Eggert. Wayne Davison designed and
implemented the unified output format. The basic algorithm is described
in "An O(ND) Difference Algorithm and its Variations", Eugene W. Myers,
'Algorithmica' Vol. 1 No. 2, 1986, pp. 251-266; and in "A File
Comparison Program", Webb Miller and Eugene W. Myers,
'Software--Practice and Experience' Vol. 15 No. 11, 1985, pp. 1025-1040.
The algorithm was independently discovered as described in "Algorithms
for Approximate String Matching", E. Ukkonen, 'Information and Control'
Vol. 64, 1985, pp. 100-118.

   GNU 'diff3' was written by Randy Smith. GNU 'sdiff' was written by
Thomas Lord. GNU 'cmp' was written by Torbjorn Granlund and David

MacKenzie.

   'patch' was written mainly by Larry Wall; the GNU enhancements were
written mainly by Wayne Davison and David MacKenzie. Parts of this
manual are adapted from a manual page written by Larry Wall, with his
permission.

1.3 diff.guide/Comparison

What Comparison Means
****

   There are several ways to think about the differences between two
files. One way to think of the differences is as a series of lines
that were deleted from, inserted in, or changed in one file to produce
the other file. 'diff' compares two files line by line, finds groups of
lines that differ, and reports each group of differing lines. It can
report the differing lines in several formats, which have different
purposes.

   GNU 'diff' can show whether files are different without detailing
the differences. It also provides ways to suppress certain kinds of
differences that are not important to you. Most commonly, such
differences are changes in the amount of white space between words or
lines. 'diff' also provides ways to suppress differences in alphabetic
case or in lines that match a regular expression that you provide.
These options can accumulate; for example, you can ignore changes in
both white space and alphabetic case.

   Another way to think of the differences between two files is as a
sequence of pairs of characters that can be either identical or
different. 'cmp' reports the differences between two files character
by character, instead of line by line. As a result, it is more useful
than 'diff' for comparing binary files. For text files, 'cmp' is
useful mainly when you want to know only whether two files are
identical.

   To illustrate the effect that considering changes character by
character can have compared with considering them line by line, think
of what happens if a single newline character is added to the beginning
of a file. If that file is then compared with an otherwise identical
file that lacks the newline at the beginning, 'diff' will report that a
blank line has been added to the file, while 'cmp' will report that
almost every character of the two files differs.

   'diff3' normally compares three input files line by line, finds
groups of lines that differ, and reports each group of differing lines.
Its output is designed to make it easy to inspect two different sets of
changes to the same file.

  Hunks   Groups of differing lines.
  White Space   Suppressing differences in white space.
  Blank Lines   Suppressing differences in blank lines.

```
Case Folding                    Suppressing differences in alphabetic case.
Specified Folding               Suppressing differences that match regular  ←
   expressions.
Brief                           Summarizing which files are different.
Binary                          Comparing binary files or forcing text comparisons.
```

1.4 diff.guide/Hunks

```
Hunks
=====
```

   When comparing two files, 'diff' finds sequences of lines common to
both files, interspersed with groups of differing lines called "hunks".
Comparing two identical files yields one sequence of common lines and
no hunks, because no lines differ. Comparing two entirely different
files yields no common lines and one large hunk that contains all lines
of both files. In general, there are many ways to match up lines
between two given files. 'diff' tries to minimize the total hunk size
by finding large sequences of common lines interspersed with small
hunks of differing lines.

   For example, suppose the file 'F' contains the three lines 'a', 'b',
'c', and the file 'G' contains the same three lines in reverse order
'c', 'b', 'a'. If 'diff' finds the line 'c' as common, then the command
'diff F G' produces this output:

```
    1,2d0
    < a
    < b
    3a2,3
    > b
    > a
```

But if 'diff' notices the common line 'b' instead, it produces this
output:

```
    1c1
    < a
    ---
    > c
    3c3
    < c
    ---
    > a
```

It is also possible to find 'a' as the common line. 'diff' does not
always find an optimal matching between the files; it takes shortcuts
to run faster. But its output is usually close to the shortest
possible. You can adjust this tradeoff with the '--minimal' option
(see diff Performance).

1.5 diff.guide/White Space

```
Suppressing Differences in Blank and Tab Spacing
================================================
```

   The '-b' and '--ignore-space-change' options ignore white space at
line end, and considers all other sequences of one or more white space
characters to be equivalent. With these options, 'diff' considers the
following two lines to be equivalent, where '$' denotes the line end:

```
    Here lyeth  muche rychnesse  in lytell space.   -- John Heywood$
    Here lyeth muche rychnesse in lytell space. -- John Heywood   $
```

   The '-w' and '--ignore-all-space' options are stronger than '-b'.
They ignore difference even if one file has white space where the other
file has none. "White space" characters include tab, newline, vertical
tab, form feed, carriage return, and space; some locales may define
additional characters to be white space. With these options, 'diff'
considers the following two lines to be equivalent, where '$' denotes
the line end and '^M' denotes a carriage return:

```
    Here lyeth  muche  rychnesse in lytell space.--  John Heywood$
      He relyeth much erychnes  seinly tells pace.  --John Heywood   ^M$
```

1.6 diff.guide/Blank Lines

```
Suppressing Differences in Blank Lines
======================================
```

   The '-B' and '--ignore-blank-lines' options ignore insertions or
deletions of blank lines. These options normally affect only lines
that are completely empty; they do not affect lines that look empty but
contain space or tab characters. With these options, for example, a
file containing
```
    1.  A point is that which has no part.

    2.  A line is breadthless length.
    -- Euclid, The Elements, I
```

is considered identical to a file containing
```
    1.  A point is that which has no part.
    2.  A line is breadthless length.


    -- Euclid, The Elements, I
```

1.7 diff.guide/Case Folding

```
Suppressing Case Differences
============================
```

GNU 'diff' can treat lowercase letters as equivalent to their
uppercase counterparts, so that, for example, it considers 'Funky
Stuff', 'funky STUFF', and 'fUNKy stuFf' to all be the same. To
request this, use the '-i' or '--ignore-case' option.

1.8   diff.guide/Specified Folding

Suppressing Lines Matching a Regular Expression
===

    To ignore insertions and deletions of lines that match a regular
expression, use the '-I REGEXP' or '--ignore-matching-lines=REGEXP'
option. You should escape regular expressions that contain shell
metacharacters to prevent the shell from expanding them. For example,
'diff -I '^[0-9]'' ignores all changes to lines beginning with a digit.

    However, '-I' only ignores the insertion or deletion of lines that
contain the regular expression if every changed line in the hunk--every
insertion and every deletion--matches the regular expression. In other
words, for each nonignorable change, 'diff' prints the complete set of
changes in its vicinity, including the ignorable ones.

    You can specify more than one regular expression for lines to ignore
by using more than one '-I' option. 'diff' tries to match each line
against each regular expression, starting with the last one given.

1.9   diff.guide/Brief

Summarizing Which Files Differ
==============================

    When you only want to find out whether files are different, and you
don't care what the differences are, you can use the summary output
format. In this format, instead of showing the differences between the
files, 'diff' simply reports whether files differ. The '-q' and
'--brief' options select this output format.

    This format is especially useful when comparing the contents of two
directories. It is also much faster than doing the normal line by line
comparisons, because 'diff' can stop analyzing the files as soon as it
knows that there are any differences.

    You can also get a brief indication of whether two files differ by
using 'cmp'. For files that are identical, 'cmp' produces no output.
When the files differ, by default, 'cmp' outputs the byte offset and
line number where the first difference occurs. You can use the '-s'
option to suppress that information, so that 'cmp' produces no output
and reports whether the files differ using only its exit status (see
Invoking cmp).

Unlike 'diff', 'cmp' cannot compare directories; it can only compare
two files.

1.10   diff.guide/Binary

Binary Files and Forcing Text Comparisons
==

   If 'diff' thinks that either of the two files it is comparing is
binary (a non-text file), it normally treats that pair of files much as
if the summary output format had been selected (see Brief), and reports
only that the binary files are different. This is because line by line
comparisons are usually not meaningful for binary files.

   'diff' determines whether a file is text or binary by checking the
first few bytes in the file; the exact number of bytes is system
dependent, but it is typically several thousand. If every character in
that part of the file is non-null, 'diff' considers the file to be
text; otherwise it considers the file to be binary.

   Sometimes you might want to force 'diff' to consider files to be
text. For example, you might be comparing text files that contain null
characters; 'diff' would erroneously decide that those are non-text
files. Or you might be comparing documents that are in a format used
by a word processing system that uses null characters to indicate
special formatting. You can force 'diff' to consider all files to be
text files, and compare them line by line, by using the '-a' or
'--text' option. If the files you compare using this option do not in
fact contain text, they will probably contain few newline characters,
and the 'diff' output will consist of hunks showing differences between
long lines of whatever characters the files contain.

   You can also force 'diff' to consider all files to be binary files,
and report only whether they differ (but not how). Use the '--brief'
option for this.

   In operating systems that distinguish between text and binary files,
'diff' normally reads and writes all data as text. Use the '--binary'
option to force 'diff' to read and write binary data instead. This
option has no effect on a Posix-compliant system like GNU or
traditional Unix. However, many personal computer operating systems
represent the end of a line with a carriage return followed by a
newline. On such systems, 'diff' normally ignores these carriage
returns on input and generates them at the end of each output line, but
with the '--binary' option 'diff' treats each carriage return as just
another input character, and does not generate a carriage return at the
end of each output line. This can be useful when dealing with non-text
files that are meant to be interchanged with Posix-compliant systems.

   If you want to compare two files byte by byte, you can use the 'cmp'
program with the '-l' option to show the values of each differing byte
in the two files. With GNU 'cmp', you can also use the '-c' option to
show the ASCII representation of those bytes. See Invoking cmp, for

more information.

   If 'diff3' thinks that any of the files it is comparing is binary (a
non-text file), it normally reports an error, because such comparisons
are usually not useful. 'diff3' uses the same test as 'diff' to decide
whether a file is binary. As with 'diff', if the input files contain a
few non-text characters but otherwise are like text files, you can
force 'diff3' to consider all files to be text files and compare them
line by line by using the '-a' or '--text' options.

1.11 diff.guide/Output Formats

'diff' Output Formats
****

   'diff' has several mutually exclusive options for output format.
The following sections describe each format, illustrating how 'diff'
reports the differences between two sample input files.

  Sample diff Input   Sample 'diff' input files for examples.
  Normal   Showing differences without surrounding text.
  Context   Showing differences with the surrounding text.
  Side by Side   Showing differences in two columns.
  Scripts   Generating scripts for other programs.
  If-then-else   Merging files with if-then-else.

1.12 diff.guide/Sample diff Input

Two Sample Input Files
======================

   Here are two sample files that we will use in numerous examples to
illustrate the output of 'diff' and how various options can change it.

   This is the file 'lao':

      The Way that can be told of is not the eternal Way;
      The name that can be named is not the eternal name.
      The Nameless is the origin of Heaven and Earth;
      The Named is the mother of all things.
      Therefore let there always be non-being,
        so we may see their subtlety,
      And let there always be being,
        so we may see their outcome.
      The two are the same,
      But after they are produced,
        they have different names.

   This is the file 'tzu':

```
    The Nameless is the origin of Heaven and Earth;
    The named is the mother of all things.

    Therefore let there always be non-being,
       so we may see their subtlety,
    And let there always be being,
       so we may see their outcome.
    The two are the same,
    But after they are produced,
       they have different names.
    They both may be called deep and profound.
    Deeper and more profound,
    The door of all subtleties!
```

   In this example, the first hunk contains just the first two lines of
'lao', the second hunk contains the fourth line of 'lao' opposing the
second and third lines of 'tzu', and the last hunk contains just the
last three lines of 'tzu'.

1.13   diff.guide/Normal

```
Showing Differences Without Context
===================================
```

   The "normal" 'diff' output format shows each hunk of differences
without any surrounding context. Sometimes such output is the clearest
way to see how lines have changed, without the clutter of nearby
unchanged lines (although you can get similar results with the context
or unified formats by using 0 lines of context). However, this format
is no longer widely used for sending out patches; for that purpose, the
context format (see Context Format) and the unified format (see
Unified Format) are superior. Normal format is the default for
compatibility with older versions of 'diff' and the Posix standard.

  Detailed Normal   A detailed description of normal output format.
  Example Normal   Sample output in the normal format.

1.14   diff.guide/Detailed Normal

```
Detailed Description of Normal Format
-------------------------------------
```

   The normal output format consists of one or more hunks of
differences; each hunk shows one area where the files differ. Normal
format hunks look like this:

```
    CHANGE-COMMAND
    < FROM-FILE-LINE
```

```
    < FROM-FILE-LINE...
    ---
    > TO-FILE-LINE
    > TO-FILE-LINE...
```

   There are three types of change commands. Each consists of a line
number or comma-separated range of lines in the first file, a single
character indicating the kind of change to make, and a line number or
comma-separated range of lines in the second file. All line numbers are
the original line numbers in each file. The types of change commands
are:

'LaR'
    Add the lines in range R of the second file after line L of the
    first file. For example, '8a12,15' means append lines 12-15 of
    file 2 after line 8 of file 1; or, if changing file 2 into file 1,
    delete lines 12-15 of file 2.

'FcT'
    Replace the lines in range F of the first file with lines in range
    T of the second file. This is like a combined add and delete, but
    more compact. For example, '5,7c8,10' means change lines 5-7 of
    file 1 to read as lines 8-10 of file 2; or, if changing file 2 into
    file 1, change lines 8-10 of file 2 to read as lines 5-7 of file 1.

'RdL'
    Delete the lines in range R from the first file; line L is where
    they would have appeared in the second file had they not been
    deleted. For example, '5,7d3' means delete lines 5-7 of file 1;
    or, if changing file 2 into file 1, append lines 5-7 of file 1
    after line 3 of file 2.

1.15   diff.guide/Example Normal

An Example of Normal Format

   Here is the output of the command 'diff lao tzu' (see
Sample diff Input, for the complete contents of the two files). Notice
that it shows only the lines that are different between the two files.

```
    1,2d0
    < The Way that can be told of is not the eternal Way;
    < The name that can be named is not the eternal name.
    4c2,3
    < The Named is the mother of all things.
    ---
    > The named is the mother of all things.
    >
    11a11,13
    > They both may be called deep and profound.
    > Deeper and more profound,
    > The door of all subtleties!
```

1.16 diff.guide/Context

```
Showing Differences in Their Context
====================================
```

Usually, when you are looking at the differences between files, you will also want to see the parts of the files near the lines that differ, to help you understand exactly what has changed. These nearby parts of the files are called the "context".

GNU 'diff' provides two output formats that show context around the differing lines: "context format" and "unified format". It can optionally show in which function or section of the file the differing lines are found.

If you are distributing new versions of files to other people in the form of 'diff' output, you should use one of the output formats that show context so that they can apply the diffs even if they have made small changes of their own to the files. 'patch' can apply the diffs in this case by searching in the files for the lines of context around the differing lines; if those lines are actually a few lines away from where the diff says they are, 'patch' can adjust the line numbers accordingly and still apply the diff correctly. See Imperfect, for more information on using 'patch' to apply imperfect diffs.

```
  Context Format          An output format that shows surrounding lines.
  Unified Format          A more compact output format that shows context.
  Sections                Showing which sections of the files differences are  ←
      in.
  Alternate Names         Showing alternate file names in context headers.
```

1.17 diff.guide/Context Format

```
Context Format
--------------
```

The context output format shows several lines of context around the lines that differ. It is the standard format for distributing updates to source code.

To select this output format, use the '-C LINES', '--context[=LINES]', or '-c' option. The argument LINES that some of these options take is the number of lines of context to show. If you do not specify LINES, it defaults to three. For proper operation, 'patch' typically needs at least two lines of context.

```
  Detailed Context        A detailed description of the context output format.
```

```
    Example Context                  Sample output in context format.
    Less Context                     Another sample with less context.
```

1.18   diff.guide/Detailed Context

```
Detailed Description of Context Format
.....................................
```

   The context output format starts with a two-line header, which looks
like this:

```
     *** FROM-FILE FROM-FILE-MODIFICATION-TIME
     --- TO-FILE TO-FILE-MODIFICATION TIME
```

You can change the header's content with the '-L LABEL' or
'--label=LABEL' option; see See Alternate Names.

   Next come one or more hunks of differences; each hunk shows one area
where the files differ. Context format hunks look like this:

```
     ***************
     *** FROM-FILE-LINE-RANGE ****
       FROM-FILE-LINE
       FROM-FILE-LINE...
     --- TO-FILE-LINE-RANGE ----
       TO-FILE-LINE
       TO-FILE-LINE...
```

   The lines of context around the lines that differ start with two
space characters. The lines that differ between the two files start
with one of the following indicator characters, followed by a space
character:

`!`
     A line that is part of a group of one or more lines that changed
     between the two files. There is a corresponding group of lines
     marked with `!` in the part of this hunk for the other file.

`+`
     An "inserted" line in the second file that corresponds to nothing
     in the first file.

`-`
     A "deleted" line in the first file that corresponds to nothing in
     the second file.

   If all of the changes in a hunk are insertions, the lines of
FROM-FILE are omitted. If all of the changes are deletions, the lines
of TO-FILE are omitted.

1.19 diff.guide/Example Context

An Example of Context Format
....

    Here is the output of 'diff -c lao tzu' (see Sample diff Input, for
the complete contents of the two files). Notice that up to three lines
that are not different are shown around each line that is different;
they are the context lines. Also notice that the first two hunks have
run together, because their contents overlap.

```
        *** lao  Sat Jan 26 23:30:39 1991
        --- tzu  Sat Jan 26 23:30:50 1991
        ***************
        *** 1,7 ****
        - The Way that can be told of is not the eternal Way;
        - The name that can be named is not the eternal name.
          The Nameless is the origin of Heaven and Earth;
        ! The Named is the mother of all things.
          Therefore let there always be non-being,
            so we may see their subtlety,
          And let there always be being,
        --- 1,6 ----
          The Nameless is the origin of Heaven and Earth;
        ! The named is the mother of all things.
        !
          Therefore let there always be non-being,
            so we may see their subtlety,
          And let there always be being,
        ***************
        *** 9,11 ****
        --- 8,13 ----
          The two are the same,
          But after they are produced,
            they have different names.
        + They both may be called deep and profound.
        + Deeper and more profound,
        + The door of all subtleties!
```

1.20 diff.guide/Less Context

An Example of Context Format with Less Context
....

    Here is the output of 'diff --context=1 lao tzu' (see
Sample diff Input, for the complete contents of the two files). Notice
that at most one context line is reported here.

```
        *** lao  Sat Jan 26 23:30:39 1991
        --- tzu  Sat Jan 26 23:30:50 1991
        ***************
        *** 1,5 ****
        - The Way that can be told of is not the eternal Way;
```

```
        - The name that can be named is not the eternal name.
          The Nameless is the origin of Heaven and Earth;
        ! The Named is the mother of all things.
          Therefore let there always be non-being,
        --- 1,4 ----
          The Nameless is the origin of Heaven and Earth;
        ! The named is the mother of all things.
        !
          Therefore let there always be non-being,
        ***************
        *** 11 ****
        --- 10,13 ----
             they have different names.
        + They both may be called deep and profound.
        + Deeper and more profound,
        + The door of all subtleties!
```

1.21   diff.guide/Unified Format

```
Unified Format
--------------
```

   The unified output format is a variation on the context format that
is more compact because it omits redundant context lines. To select
this output format, use the '-U LINES', '--unified[=LINES]', or '-u'
option. The argument LINES is the number of lines of context to show.
When it is not given, it defaults to three.

   At present, only GNU 'diff' can produce this format and only GNU
'patch' can automatically apply diffs in this format. For proper
operation, 'patch' typically needs at least two lines of context.

  Detailed Unified   A detailed description of unified format.
  Example Unified   Sample output in unified format.

1.22   diff.guide/Detailed Unified

```
Detailed Description of Unified Format
......................................
```

   The unified output format starts with a two-line header, which looks
like this:

```
     --- FROM-FILE FROM-FILE-MODIFICATION-TIME
     +++ TO-FILE TO-FILE-MODIFICATION-TIME
```

You can change the header's content with the '-L LABEL' or
'--label=LABEL' option; see See Alternate Names.

Next come one or more hunks of differences; each hunk shows one area
where the files differ. Unified format hunks look like this:

```
@@ FROM-FILE-RANGE TO-FILE-RANGE @@
  LINE-FROM-EITHER-FILE
  LINE-FROM-EITHER-FILE...
```

The lines common to both files begin with a space character. The
lines that actually differ between the two files have one of the
following indicator characters in the left column:

`+`
    A line was added here to the first file.

`-`
    A line was removed here from the first file.

1.23   diff.guide/Example Unified

An Example of Unified Format
....

Here is the output of the command 'diff -u lao tzu' (see
Sample diff Input, for the complete contents of the two files):

```
--- lao  Sat Jan 26 23:30:39 1991
+++ tzu  Sat Jan 26 23:30:50 1991
@@ -1,7 +1,6 @@
-The Way that can be told of is not the eternal Way;
-The name that can be named is not the eternal name.
 The Nameless is the origin of Heaven and Earth;
-The Named is the mother of all things.
+The named is the mother of all things.
+
 Therefore let there always be non-being,
   so we may see their subtlety,
 And let there always be being,
@@ -9,3 +8,6 @@
 The two are the same,
 But after they are produced,
   they have different names.
+They both may be called deep and profound.
+Deeper and more profound,
+The door of all subtleties!
```


1.24   diff.guide/Sections

Showing Which Sections Differences Are in

Sometimes you might want to know which part of the files each change falls in. If the files are source code, this could mean which function was changed. If the files are documents, it could mean which chapter or appendix was changed. GNU 'diff' can show this by displaying the nearest section heading line that precedes the differing lines. Which lines are "section headings" is determined by a regular expression.

Specified Headings   Showing headings that match regular expressions.
C Function Headings   Showing headings of C functions.

1.25   diff.guide/Specified Headings

Showing Lines That Match Regular Expressions
....

   To show in which sections differences occur for files that are not source code for C or similar languages, use the '-F REGEXP' or '--show-function-line=REGEXP' option. 'diff' considers lines that match the argument REGEXP to be the beginning of a section of the file. Here are suggested regular expressions for some common languages:

'^[A-Za-z_]'
    C, C++, Prolog

'^('
    Lisp

'^@\(chapter\|appendix\|unnumbered\|chapheading\)'
    Texinfo

   This option does not automatically select an output format; in order to use it, you must select the context format (see Context Format) or unified format (see Unified Format). In other output formats it has no effect.

   The '-F' and '--show-function-line' options find the nearest unchanged line that precedes each hunk of differences and matches the given regular expression. Then they add that line to the end of the line of asterisks in the context format, or to the '@@' line in unified format. If no matching line exists, they leave the output for that hunk unchanged. If that line is more than 40 characters long, they output only the first 40 characters. You can specify more than one regular expression for such lines; 'diff' tries to match each line against each regular expression, starting with the last one given. This means that you can use '-p' and '-F' together, if you wish.

1.26   diff.guide/C Function Headings

Showing C Function Headings
....

   To show in which functions differences occur for C and similar
languages, you can use the '-p' or '--show-c-function' option. This
option automatically defaults to the context output format (see
Context Format), with the default number of lines of context. You can
override that number with '-C LINES' elsewhere in the command line.
You can override both the format and the number with '-U LINES'
elsewhere in the command line.

   The '-p' and '--show-c-function' options are equivalent to
'-F'^[_a-zA-Z$]'' if the unified format is specified, otherwise '-c
-F'^[_a-zA-Z$]'' (see Specified Headings). GNU 'diff' provides them
for the sake of convenience.

1.27   diff.guide/Alternate Names

Showing Alternate File Names

   If you are comparing two files that have meaningless or uninformative
names, you might want 'diff' to show alternate names in the header of
the context and unified output formats. To do this, use the '-L LABEL'
or '--label=LABEL' option. The first time you give this option, its
argument replaces the name and date of the first file in the header;
the second time, its argument replaces the name and date of the second
file. If you give this option more than twice, 'diff' reports an
error. The '-L' option does not affect the file names in the 'pr'
header when the '-l' or '--paginate' option is used (see Pagination).

   Here are the first two lines of the output from 'diff -C2 -Loriginal
-Lmodified lao tzu':

      *** original
      --- modified

1.28   diff.guide/Side by Side

Showing Differences Side by Side
================================

   'diff' can produce a side by side difference listing of two files.
The files are listed in two columns with a gutter between them. The
gutter contains one of the following markers:

white space
      The corresponding lines are in common. That is, either the lines
      are identical, or the difference is ignored because of one of the

'--ignore' options (see White Space).

'|'

The corresponding lines differ, and they are either both complete
or both incomplete.

'<'

The files differ and only the first file contains the line.

'>'

The files differ and only the second file contains the line.

'('

Only the first file contains the line, but the difference is
ignored.

')'

Only the second file contains the line, but the difference is
ignored.

'\'

The corresponding lines differ, and only the first line is
incomplete.

'/'

The corresponding lines differ, and only the second line is
incomplete.

Normally, an output line is incomplete if and only if the lines that
it contains are incomplete; See Incomplete Lines. However, when an
output line represents two differing lines, one might be incomplete
while the other is not. In this case, the output line is complete, but
its the gutter is marked '\' if the first line is incomplete, '/' if
the second line is.

Side by side format is sometimes easiest to read, but it has
limitations. It generates much wider output than usual, and truncates
lines that are too long to fit. Also, it relies on lining up output
more heavily than usual, so its output looks particularly bad if you
use varying width fonts, nonstandard tab stops, or nonprinting
characters.

You can use the 'sdiff' command to interactively merge side by side
differences. See Interactive Merging, for more information on merging
files.

Side by Side Format   Controlling side by side output format.
Example Side by Side   Sample side by side output.

1.29   diff.guide/Side by Side Format

Controlling Side by Side Format
===============================

The '-y' or '--side-by-side' option selects side by side format. Because side by side output lines contain two input lines, they are wider than usual. They are normally 130 columns, which can fit onto a traditional printer line. You can set the length of output lines with the '-W COLUMNS' or '--width=COLUMNS' option. The output line is split into two halves of equal length, separated by a small gutter to mark differences; the right half is aligned to a tab stop so that tabs line up. Input lines that are too long to fit in half of an output line are truncated for output.

The '--left-column' option prints only the left column of two common lines. The '--suppress-common-lines' option suppresses common lines entirely.

1.30 diff.guide/Example Side by Side

An Example of Side by Side Format

Here is the output of the command 'diff -y -W 72 lao tzu' (see Sample diff Input, for the complete contents of the two files).

```
    The Way that can be told of is n  <
    The name that can be named is no  <
    The Nameless is the origin of He       The Nameless is the origin of He
    The Named is the mother of all t  |    The named is the mother of all t
                                      >
    Therefore let there always be no       Therefore let there always be no
      so we may see their subtlety,           so we may see their subtlety,
    And let there always be being,         And let there always be being,
      so we may see their outcome.            so we may see their outcome.
    The two are the same,                  The two are the same,
    But after they are produced,           But after they are produced,
      they have different names.              they have different names.
                                      >    They both may be called deep and
                                      >    Deeper and more profound,
                                      >    The door of all subtleties!
```


1.31 diff.guide/Scripts

Making Edit Scripts
===================

Several output modes produce command scripts for editing FROM-FILE to produce TO-FILE.


```
  ed Scripts                Using 'diff' to produce commands for 'ed'.
  Forward ed                Making forward 'ed' scripts.
```

RCS   A special 'diff' output format used by RCS.

1.32 diff.guide/ed Scripts

'ed' Scripts

    'diff' can produce commands that direct the 'ed' text editor to
change the first file into the second file. Long ago, this was the
only output mode that was suitable for editing one file into another
automatically; today, with 'patch', it is almost obsolete. Use the
'-e' or '--ed' option to select this output format.

    Like the normal format (see Normal), this output format does not
show any context; unlike the normal format, it does not include the
information necessary to apply the diff in reverse (to produce the first
file if all you have is the second file and the diff).

    If the file 'd' contains the output of 'diff -e old new', then the
command '(cat d && echo w) | ed - old' edits 'old' to make it a copy of
'new'. More generally, if 'd1', 'd2', ..., 'dN' contain the outputs of
'diff -e old new1', 'diff -e new1 new2', ..., 'diff -e newN-1 newN',
respectively, then the command '(cat d1 d2 ... dN && echo w) | ed -
old' edits 'old' to make it a copy of 'newN'.

  Detailed ed   A detailed description of 'ed' format.
  Example ed   A sample 'ed' script.

1.33 diff.guide/Detailed ed

Detailed Description of 'ed' Format
....

    The 'ed' output format consists of one or more hunks of differences.
The changes closest to the ends of the files come first so that
commands that change the number of lines do not affect how 'ed'
interprets line numbers in succeeding commands. 'ed' format hunks look
like this:

    CHANGE-COMMAND
    TO-FILE-LINE
    TO-FILE-LINE...
    .

    Because 'ed' uses a single period on a line to indicate the end of
input, GNU 'diff' protects lines of changes that contain a single
period on a line by writing two periods instead, then writing a
subsequent 'ed' command to change the two periods into one. The 'ed'
format cannot represent an incomplete line, so if the second file ends

in a changed incomplete line, 'diff' reports an error and then pretends
that a newline was appended.

   There are three types of change commands. Each consists of a line
number or comma-separated range of lines in the first file and a single
character indicating the kind of change to make. All line numbers are
the original line numbers in the file. The types of change commands
are:

'La'
     Add text from the second file after line L in the first file. For
     example, '8a' means to add the following lines after line 8 of file
     1.

'Rc'
     Replace the lines in range R in the first file with the following
     lines. Like a combined add and delete, but more compact. For
     example, '5,7c' means change lines 5-7 of file 1 to read as the
     text file 2.

'Rd'
     Delete the lines in range R from the first file. For example,
     '5,7d' means delete lines 5-7 of file 1.

1.34   diff.guide/Example ed

Example 'ed' Script
....

   Here is the output of 'diff -e lao tzu' (see Sample diff Input, for
the complete contents of the two files):

     11a
     They both may be called deep and profound.
     Deeper and more profound,
     The door of all subtleties!
     .
     4c
     The named is the mother of all things.

     .
     1,2d

1.35   diff.guide/Forward ed

Forward 'ed' Scripts

   'diff' can produce output that is like an 'ed' script, but with
hunks in forward (front to back) order. The format of the commands is

also changed slightly: command characters precede the lines they
modify, spaces separate line numbers in ranges, and no attempt is made
to disambiguate hunk lines consisting of a single period. Like 'ed'
format, forward 'ed' format cannot represent incomplete lines.

    Forward 'ed' format is not very useful, because neither 'ed' nor
'patch' can apply diffs in this format. It exists mainly for
compatibility with older versions of 'diff'. Use the '-f' or
'--forward-ed' option to select it.

1.36   diff.guide/RCS

RCS Scripts

    The RCS output format is designed specifically for use by the
Revision Control System, which is a set of free programs used for
organizing different versions and systems of files. Use the '-n' or
'--rcs' option to select this output format. It is like the forward
'ed' format (see Forward ed), but it can represent arbitrary changes to
the contents of a file because it avoids the forward 'ed' format's
problems with lines consisting of a single period and with incomplete
lines. Instead of ending text sections with a line consisting of a
single period, each command specifies the number of lines it affects; a
combination of the 'a' and 'd' commands are used instead of 'c'. Also,
if the second file ends in a changed incomplete line, then the output
also ends in an incomplete line.

    Here is the output of 'diff -n lao tzu' (see Sample diff Input, for
the complete contents of the two files):

    d1 2
    d4 1
    a4 2
    The named is the mother of all things.

    a11 3
    They both may be called deep and profound.
    Deeper and more profound,
    The door of all subtleties!

1.37   diff.guide/If-then-else

Merging Files with If-then-else
===============================

    You can use 'diff' to merge two files of C source code. The output
of 'diff' in this format contains all the lines of both files. Lines
common to both files are output just once; the differing parts are
separated by the C preprocessor directives '#ifdef NAME' or '#ifndef

NAME', '#else', and '#endif'. When compiling the output, you select
which version to use by either defining or leaving undefined the macro
NAME.

   To merge two files, use 'diff' with the '-D NAME' or '--ifdef=NAME'
option. The argument NAME is the C preprocessor identifier to use in
the '#ifdef' and '#ifndef' directives.

   For example, if you change an instance of 'wait (&s)' to 'waitpid
(-1, &s, 0)' and then merge the old and new files with the
'--ifdef=HAVE_WAITPID' option, then the affected part of your code
might look like this:

```
        do {
    #ifndef HAVE_WAITPID
            if ((w = wait (&s)) < 0  &&  errno != EINTR)
    #else /* HAVE_WAITPID */
            if ((w = waitpid (-1, &s, 0)) < 0  &&  errno != EINTR)
    #endif /* HAVE_WAITPID */
                return w;
        } while (w != child);
```

   You can specify formats for languages other than C by using line
group formats and line formats, as described in the next sections.

1.38   diff.guide/Line Group Formats

Line Group Formats

   Line group formats let you specify formats suitable for many
applications that allow if-then-else input, including programming
languages and text formatting languages. A line group format specifies
the output format for a contiguous group of similar lines.

   For example, the following command compares the TeX files 'old' and
'new', and outputs a merged file in which old regions are surrounded by
'\begin{em}'-'\end{em}' lines, and new regions are surrounded by
'\begin{bf}'-'\end{bf}' lines.

```
    diff \
       --old-group-format='\begin{em}
    %<\end{em}
    ' \
       --new-group-format='\begin{bf}
    %>\end{bf}
    ' \
       old new
```

The following command is equivalent to the above example, but it is a
little more verbose, because it spells out the default line group
formats.

```
diff \
    --old-group-format='\begin{em}
%<\end{em}
' \
    --new-group-format='\begin{bf}
%>\end{bf}
' \
    --unchanged-group-format='%=' \
    --changed-group-format='\begin{em}
%<\end{em}
\begin{bf}
%>\end{bf}
' \
    old new
```

Here is a more advanced example, which outputs a diff listing with
headers containing line numbers in a "plain English" style.

```
diff \
    --unchanged-group-format='' \
    --old-group-format='-------- %dn line%(n=1?:s) deleted at %df:
%<' \
    --new-group-format='-------- %dN line%(N=1?:s) added after %de:
%>' \
    --changed-group-format='-------- %dn line%(n=1?:s) changed at %df:
%<-------- to:
%>' \
    old new
```

To specify a line group format, use 'diff' with one of the options
listed below. You can specify up to four line group formats, one for
each kind of line group. You should quote FORMAT, because it typically
contains shell metacharacters.

'--old-group-format=FORMAT'
    These line groups are hunks containing only lines from the first
    file. The default old group format is the same as the changed
    group format if it is specified; otherwise it is a format that
    outputs the line group as-is.

'--new-group-format=FORMAT'
    These line groups are hunks containing only lines from the second
    file. The default new group format is same as the the changed
    group format if it is specified; otherwise it is a format that
    outputs the line group as-is.

'--changed-group-format=FORMAT'
    These line groups are hunks containing lines from both files. The
    default changed group format is the concatenation of the old and
    new group formats.

'--unchanged-group-format=FORMAT'

These line groups contain lines common to both files. The default
unchanged group format is a format that outputs the line group
as-is.

   In a line group format, ordinary characters represent themselves;
conversion specifications start with '%' and have one of the following
forms.

'%<'

     stands for the lines from the first file, including the trailing
     newline. Each line is formatted according to the old line format
     (see Line Formats).

'%>'

     stands for the lines from the second file, including the trailing
     newline. Each line is formatted according to the new line format.

'%='

     stands for the lines common to both files, including the trailing
     newline. Each line is formatted according to the unchanged line
     format.

'%%'

     stands for '%'.

'%c'C''

     where C is a single character, stands for C. C may not be a
     backslash or an apostrophe. For example, '%c':'' stands for a
     colon, even inside the then-part of an if-then-else format, which
     a colon would normally terminate.

'%c'\O''

     where O is a string of 1, 2, or 3 octal digits, stands for the
     character with octal code O. For example, '%c'\0'' stands for a
     null character.

'FN'

     where F is a 'printf' conversion specification and N is one of the
     following letters, stands for N's value formatted with F.

   'e'

       The line number of the line just before the group in the old
       file.

   'f'

       The line number of the first line in the group in the old
       file; equals E + 1.

   'l'

       The line number of the last line in the group in the old file.

   'm'

       The line number of the line just after the group in the old
       file; equals L + 1.

   'n'

       The number of lines in the group in the old file; equals L –

```
           F + 1.

    'E, F, L, M, N'
            Likewise, for lines in the new file.

    The 'printf' conversion specification can be '%d', '%o', '%x', or
    '%X', specifying decimal, octal, lower case hexadecimal, or upper
    case hexadecimal output respectively.  After the '%' the following
    options can appear in sequence: a '-' specifying
    left-justification; an integer specifying the minimum field width;
    and a period followed by an optional integer specifying the
    minimum number of digits.  For example, '%5dN' prints the number
    of new lines in the group in a field of width 5 characters, using
    the 'printf' format '"%5d"'.

'(A=B?T:E)'
    If A equals B then T else E.  A and B are each either a decimal
    constant or a single letter interpreted as above.  This format
    spec is equivalent to T if A's value equals B's; otherwise it is
    equivalent to E.

    For example, '%(N=0?no:%dN) line%(N=1?:s)' is equivalent to 'no
    lines' if N (the number of lines in the group in the the new file)
    is 0, to '1 line' if N is 1, and to '%dN lines' otherwise.
```

1.39 diff.guide/Line Formats

```
Line Formats
------------

   Line formats control how each line taken from an input file is
output as part of a line group in if-then-else format.

   For example, the following command outputs text with a one-column
change indicator to the left of the text.  The first column of output
is '-' for deleted lines, '|' for added lines, and a space for
unchanged lines.  The formats contain newline characters where newlines
are desired on output.

        diff \
           --old-line-format='-%l
        ' \
           --new-line-format='|%l
        ' \
           --unchanged-line-format=' %l
        ' \
           old new

   To specify a line format, use one of the following options.  You
should quote FORMAT, since it often contains shell metacharacters.

'--old-line-format=FORMAT'
    formats lines just from the first file.
```

'--new-line-format=FORMAT'
     formats lines just from the second file.

'--unchanged-line-format=FORMAT'
     formats lines common to both files.

'--line-format=FORMAT'
     formats all lines; in effect, it sets all three above options
     simultaneously.

     In a line format, ordinary characters represent themselves;
conversion specifications start with '%' and have one of the following
forms.

'%l'
     stands for the the contents of the line, not counting its trailing
     newline (if any). This format ignores whether the line is
     incomplete; See Incomplete Lines.

'%L'
     stands for the the contents of the line, including its trailing
     newline (if any). If a line is incomplete, this format preserves
     its incompleteness.

'%%'
     stands for '%'.

'%c'C''
     where C is a single character, stands for C. C may not be a
     backslash or an apostrophe. For example, '%c':'' stands for a
     colon.

'%c'\O''
     where O is a string of 1, 2, or 3 octal digits, stands for the
     character with octal code O. For example, '%c'\0'' stands for a
     null character.

'Fn'
     where F is a 'printf' conversion specification, stands for the
     line number formatted with F. For example, '%.5dn' prints the
     line number using the 'printf' format '"%.5d"'. See
     Line Group Formats, for more about printf conversion
     specifications.

     The default line format is '%l' followed by a newline character.

     If the input contains tab characters and it is important that they
line up on output, you should ensure that '%l' or '%L' in a line format
is just after a tab stop (e.g. by preceding '%l' or '%L' with a tab
character), or you should use the '-t' or '--expand-tabs' option.

     Taken together, the line and line group formats let you specify many
different formats. For example, the following command uses a format
similar to 'diff''s normal format. You can tailor this command to get
fine control over 'diff''s output.

     diff \

```
    --old-line-format='< %l
' \
    --new-line-format='> %l
' \
    --old-group-format='%df%(f=l?:,%dl)d%dE
%<' \
    --new-group-format='%dea%dF%(F=L?:,%dL)
%>' \
    --changed-group-format='%df%(f=l?:,%dl)c%dF%(F=L?:,%dL)
%<---
%>' \
    --unchanged-group-format='' \
    old new
```

1.40   diff.guide/Detailed If-then-else

```
Detailed Description of If-then-else Format
-------------------------------------------
```

   For lines common to both files, 'diff' uses the unchanged line group
format. For each hunk of differences in the merged output format, if
the hunk contains only lines from the first file, 'diff' uses the old
line group format; if the hunk contains only lines from the second
file, 'diff' uses the new group format; otherwise, 'diff' uses the
changed group format.

   The old, new, and unchanged line formats specify the output format of
lines from the first file, lines from the second file, and lines common
to both files, respectively.

   The option '--ifdef=NAME' is equivalent to the following sequence of
options using shell syntax:

```
    --old-group-format='#ifndef NAME
%<#endif /* not NAME */
' \
    --new-group-format='#ifdef NAME
%>#endif /* NAME */
' \
    --unchanged-group-format='%=' \
    --changed-group-format='#ifndef NAME
%<#else /* NAME */
%>#endif /* NAME */
'
```

   You should carefully check the 'diff' output for proper nesting.
For example, when using the the '-D NAME' or '--ifdef=NAME' option, you
should check that if the differing lines contain any of the C
preprocessor directives '#ifdef', '#ifndef', '#else', '#elif', or
'#endif', they are nested properly and match. If they don't, you must
make corrections manually. It is a good idea to carefully check the
resulting code anyway to make sure that it really does what you want it
to; depending on how the input files were produced, the output might
contain duplicate or otherwise incorrect code.

The 'patch' '-D NAME' option behaves just like the 'diff' '-D NAME'
option, except it operates on a file and a diff to produce a merged
file; See patch Options.

1.41   diff.guide/Example If-then-else

An Example of If-then-else Format

   Here is the output of 'diff -DTWO lao tzu' (see Sample diff Input,
for the complete contents of the two files):

```
    #ifndef TWO
    The Way that can be told of is not the eternal Way;
    The name that can be named is not the eternal name.
    #endif /* not TWO */
    The Nameless is the origin of Heaven and Earth;
    #ifndef TWO
    The Named is the mother of all things.
    #else /* TWO */
    The named is the mother of all things.

    #endif /* TWO */
    Therefore let there always be non-being,
      so we may see their subtlety,
    And let there always be being,
      so we may see their outcome.
    The two are the same,
    But after they are produced,
      they have different names.
    #ifdef TWO
    They both may be called deep and profound.
    Deeper and more profound,
    The door of all subtleties!
    #endif /* TWO */
```


1.42   diff.guide/Comparing Directories

Comparing Directories
****

   You can use 'diff' to compare some or all of the files in two
directory trees. When both file name arguments to 'diff' are
directories, it compares each file that is contained in both
directories, examining file names in alphabetical order. Normally
'diff' is silent about pairs of files that contain no differences, but
if you use the '-s' or '--report-identical-files' option, it reports
pairs of identical files. Normally 'diff' reports subdirectories
common to both directories without comparing subdirectories' files, but

if you use the '-r' or '--recursive' option, it compares every
corresponding pair of files in the directory trees, as many levels deep
as they go.

   For file names that are in only one of the directories, 'diff'
normally does not show the contents of the file that exists; it reports
only that the file exists in that directory and not in the other. You
can make 'diff' act as though the file existed but was empty in the
other directory, so that it outputs the entire contents of the file that
actually exists. (It is output as either an insertion or a deletion,
depending on whether it is in the first or the second directory given.)
To do this, use the '-N' or '--new-file' option.

   If the older directory contains one or more large files that are not
in the newer directory, you can make the patch smaller by using the
'-P' or '--unidirectional-new-file' option instead of '-N'. This
option is like '-N' except that it only inserts the contents of files
that appear in the second directory but not the first (that is, files
that were added). At the top of the patch, write instructions for the
user applying the patch to remove the files that were deleted before
applying the patch. See Making Patches, for more discussion of making
patches for distribution.

   To ignore some files while comparing directories, use the '-x
PATTERN' or '--exclude=PATTERN' option. This option ignores any files
or subdirectories whose base names match the shell pattern PATTERN.
Unlike in the shell, a period at the start of the base of a file name
matches a wildcard at the start of a pattern. You should enclose
PATTERN in quotes so that the shell does not expand it. For example,
the option '-x '*.[ao]'' ignores any file whose name ends with '.a' or
'.o'.

   This option accumulates if you specify it more than once. For
example, using the options '-x 'RCS' -x '*,v'' ignores any file or
subdirectory whose base name is 'RCS' or ends with ',v'.

   If you need to give this option many times, you can instead put the
patterns in a file, one pattern per line, and use the '-X FILE' or
'--exclude-from=FILE' option.

   If you have been comparing two directories and stopped partway
through, later you might want to continue where you left off. You can
do this by using the '-S FILE' or '--starting-file=FILE' option. This
compares only the file FILE and all alphabetically later files in the
topmost directory level.

1.43 diff.guide/Adjusting Output

Making 'diff' Output Prettier
****

   'diff' provides several ways to adjust the appearance of its output.
These adjustments can be applied to any output format.

        Tabs   Preserving the alignment of tabstops.
        Pagination   Page numbering and timestamping `diff' output.

1.44   diff.guide/Tabs

Preserving Tabstop Alignment
============================

   The lines of text in some of the `diff' output formats are preceded
by one or two characters that indicate whether the text is inserted,
deleted, or changed. The addition of those characters can cause tabs to
move to the next tabstop, throwing off the alignment of columns in the
line. GNU `diff' provides two ways to make tab-aligned columns line up
correctly.

   The first way is to have `diff' convert all tabs into the correct
number of spaces before outputting them; select this method with the
`-t' or `--expand-tabs' option. `diff' assumes that tabstops are set
every 8 columns. To use this form of output with `patch', you must
give `patch' the `-l' or `--ignore-white-space' option (see
Changed White Space, for more information).

   The other method for making tabs line up correctly is to add a tab
character instead of a space after the indicator character at the
beginning of the line. This ensures that all following tab characters
are in the same position relative to tabstops that they were in the
original files, so that the output is aligned correctly. Its
disadvantage is that it can make long lines too long to fit on one line
of the screen or the paper. It also does not work with the unified
output format, which does not have a space character after the change
type indicator character. Select this method with the `-T' or
`--initial-tab' option.

1.45   diff.guide/Pagination

Paginating `diff' Output
========================

   It can be convenient to have long output page-numbered and
time-stamped. The `-l' and `--paginate' options do this by sending the
`diff' output through the `pr' program. Here is what the page header
might look like for `diff -lc lao tzu':

     Mar 11 13:37 1991 diff -lc lao tzu Page 1

1.46   diff.guide/diff Performance

'diff' Performance Tradeoffs
****

    GNU 'diff' runs quite efficiently; however, in some circumstances
you can cause it to run faster or produce a more compact set of changes.
There are two ways that you can affect the performance of GNU 'diff' by
changing the way it compares files.

    Performance has more than one dimension. These options improve one
aspect of performance at the cost of another, or they improve
performance in some cases while hurting it in others.

    The way that GNU 'diff' determines which lines have changed always
comes up with a near-minimal set of differences. Usually it is good
enough for practical purposes. If the 'diff' output is large, you
might want 'diff' to use a modified algorithm that sometimes produces a
smaller set of differences. The '-d' or '--minimal' option does this;
however, it can also cause 'diff' to run more slowly than usual, so it
is not the default behavior.

    When the files you are comparing are large and have small groups of
changes scattered throughout them, you can use the '-H' or
'--speed-large-files' option to make a different modification to the
algorithm that 'diff' uses. If the input files have a constant small
density of changes, this option speeds up the comparisons without
changing the output. If not, 'diff' might produce a larger set of
differences; however, the output will still be correct.

    Normally 'diff' discards the prefix and suffix that is common to
both files before it attempts to find a minimal set of differences.
This makes 'diff' run faster, but occasionally it may produce
non-minimal output. The '--horizon-lines=LINES' option prevents 'diff'
from discarding the last LINES lines of the prefix and the first LINES
lines of the suffix. This gives 'diff' further opportunities to find a
minimal output.

1.47   diff.guide/Comparing Three Files

Comparing Three Files
****

    Use the program 'diff3' to compare three files and show any
differences among them. ('diff3' can also merge files; see See
diff3 Merging.).

    The "normal" 'diff3' output format shows each hunk of differences
without surrounding context. Hunks are labeled depending on whether
they are two-way or three-way, and lines are annotated by their
location in the input files.

    See Invoking diff3, for more information on how to run 'diff3'.

1.48 diff.guide/Sample diff3 Input

```
A Third Sample Input File
=========================
```

   Here is a third sample file that will be used in examples to
illustrate the output of 'diff3' and how various options can change it.
The first two files are the same that we used for 'diff' (see
Sample diff Input). This is the third sample file, called 'tao':

```
    The Way that can be told of is not the eternal Way;
    The name that can be named is not the eternal name.
    The Nameless is the origin of Heaven and Earth;
    The named is the mother of all things.

    Therefore let there always be non-being,
      so we may see their subtlety,
    And let there always be being,
      so we may see their result.
    The two are the same,
    But after they are produced,
      they have different names.

        -- The Way of Lao-Tzu, tr. Wing-tsit Chan
```

1.49 diff.guide/Detailed diff3 Normal

```
Detailed Description of 'diff3' Normal Format
=============================================
```

   Each hunk begins with a line marked '===='. Three-way hunks have
plain '====' lines, and two-way hunks have '1', '2', or '3' appended to
specify which of the three input files differ in that hunk. The hunks
contain copies of two or three sets of input lines each preceded by one
or two commands identifying where the lines came from.

   Normally, two spaces precede each copy of an input line to
distinguish it from the commands. But with the '-T' or '--initial-tab'
option, 'diff3' uses a tab instead of two spaces; this lines up tabs
correctly. See Tabs, for more information.

   Commands take the following forms:

'FILE:La'
     This hunk appears after line L of file FILE, and contains no lines

in that file. To edit this file to yield the other files, one
must append hunk lines taken from the other files. For example,
`1:11a' means that the hunk follows line 11 in the first file and
contains no lines from that file.

`FILE:Rc'
     This hunk contains the lines in the range R of file FILE. The
     range R is a comma-separated pair of line numbers, or just one
     number if the range is a singleton. To edit this file to yield the
     other files, one must change the specified lines to be the lines
     taken from the other files. For example, `2:11,13c' means that
     the hunk contains lines 11 through 13 from the second file.

     If the last line in a set of input lines is incomplete (see
Incomplete Lines), it is distinguished on output from a full line by a
following line that starts with `\'.

1.50 diff.guide/diff3 Hunks

`diff3' Hunks
============

     Groups of lines that differ in two or three of the input files are
called "diff3 hunks", by analogy with `diff' hunks (see Hunks). If all
three input files differ in a `diff3' hunk, the hunk is called a
"three-way hunk"; if just two input files differ, it is a "two-way
hunk".

     As with `diff', several solutions are possible. When comparing the
files `A', `B', and `C', `diff3' normally finds `diff3' hunks by
merging the two-way hunks output by the two commands `diff A B' and
`diff A C'. This does not necessarily minimize the size of the output,
but exceptions should be rare.

     For example, suppose `F' contains the three lines `a', `b', `f', `G'
contains the lines `g', `b', `g', and `H' contains the lines `a', `b',
`h'.  `diff3 F G H' might output the following:

     ====2
     1:1c
     3:1c
       a
     2:1c
       g
     ====
     1:3c
       f
     2:3c
       g
     3:3c
       h

because it found a two-way hunk containing `a' in the first and third
files and `g' in the second file, then the single line `b' common to

all three files, then a three-way hunk containing the last line of each
file.

1.51   diff.guide/Example diff3 Normal

```
An Example of 'diff3' Normal Format
===================================
```

   Here is the output of the command 'diff3 lao tzu tao' (see
Sample diff3 Input, for the complete contents of the files). Notice
that it shows only the lines that are different among the three files.

```
    ====2
    1:1,2c
    3:1,2c
      The Way that can be told of is not the eternal Way;
      The name that can be named is not the eternal name.
    2:0a
    ====1
    1:4c
      The Named is the mother of all things.
    2:2,3c
    3:4,5c
      The named is the mother of all things.

    ====3
    1:8c
    2:7c
        so we may see their outcome.
    3:9c
        so we may see their result.
    ====
    1:11a
    2:11,13c
      They both may be called deep and profound.
      Deeper and more profound,
      The door of all subtleties!
    3:13,14c

        -- The Way of Lao-Tzu, tr. Wing-tsit Chan
```

1.52   diff.guide/diff3 Merging

```
Merging From a Common Ancestor
******************************
```

   When two people have made changes to copies of the same file,
'diff3' can produce a merged output that contains both sets of changes
together with warnings about conflicts.

   One might imagine programs with names like 'diff4' and 'diff5' to
compare more than three files simultaneously, but in practice the need
rarely arises. You can use 'diff3' to merge three or more sets of
changes to a file by merging two change sets at a time.

   'diff3' can incorporate changes from two modified versions into a
common preceding version. This lets you merge the sets of changes
represented by the two newer files. Specify the common ancestor version
as the second argument and the two newer versions as the first and third
arguments, like this:

     diff3 MINE OLDER YOURS

You can remember the order of the arguments by noting that they are in
alphabetical order.

   You can think of this as subtracting OLDER from YOURS and adding the
result to MINE, or as merging into MINE the changes that would turn
OLDER into YOURS. This merging is well-defined as long as MINE and
OLDER match in the neighborhood of each such change. This fails to be
true when all three input files differ or when only OLDER differs; we
call this a "conflict". When all three input files differ, we call the
conflict an "overlap".

   'diff3' gives you several ways to handle overlaps and conflicts.
You can omit overlaps or conflicts, or select only overlaps, or mark
conflicts with special '<<<<<<<' and '>>>>>>>' lines.

   'diff3' can output the merge results as an 'ed' script that that can
be applied to the first file to yield the merged output. However, it
is usually better to have 'diff3' generate the merged output directly;
this bypasses some problems with 'ed'.

  Which Changes   Selecting changes to incorporate.
  Marking Conflicts   Marking conflicts.
  Bypassing ed   Generating merged output directly.
  Merging Incomplete Lines   How 'diff3' merges incomplete lines.
  Saving the Changed File   Emulating System V behavior.

1.53   diff.guide/Which Changes

Selecting Which Changes to Incorporate
======================================

   You can select all unmerged changes from OLDER to YOURS for merging
into MINE with the '-e' or '--ed' option. You can select only the
nonoverlapping unmerged changes with '-3' or '--easy-only', and you can
select only the overlapping changes with '-x' or '--overlap-only'.

   The '-e', '-3' and '-x' options select only "unmerged changes", i.e.
changes where MINE and YOURS differ; they ignore changes from OLDER to
YOURS where MINE and YOURS are identical, because they assume that such
changes have already been merged. If this assumption is not a safe

one, you can use the '-A' or '--show-all' option (see
Marking Conflicts).

   Here is the output of the command 'diff3' with each of these three
options (see Sample diff3 Input, for the complete contents of the
files). Notice that '-e' outputs the union of the disjoint sets of
changes output by '-3' and '-x'.

     Output of 'diff3 -e lao tzu tao':
       11a

         -- The Way of Lao-Tzu, tr. Wing-tsit Chan
       .
       8c
         so we may see their result.
       .

     Output of 'diff3 -3 lao tzu tao':
       8c
         so we may see their result.
       .

     Output of 'diff3 -x lao tzu tao':
       11a

         -- The Way of Lao-Tzu, tr. Wing-tsit Chan
       .

1.54 diff.guide/Marking Conflicts

Marking Conflicts
=================

   'diff3' can mark conflicts in the merged output by bracketing them
with special marker lines. A conflict that comes from two files A and
B is marked as follows:

     <<<<<<< A
     lines from A
     =======
     lines from B
     >>>>>>> B

   A conflict that comes from three files A, B and C is marked as
follows:

     <<<<<<< A
     lines from A
     ||||||| B
     lines from B
     =======
     lines from C
     >>>>>>> C

The '-A' or '--show-all' option acts like the '-e' option, except
that it brackets conflicts, and it outputs all changes from OLDER to
YOURS, not just the unmerged changes. Thus, given the sample input
files (see Sample diff3 Input), 'diff3 -A lao tzu tao' puts brackets
around the conflict where only 'tzu' differs:

```
<<<<<<< tzu
=======
The Way that can be told of is not the eternal Way;
The name that can be named is not the eternal name.
>>>>>>> tao
```

And it outputs the three-way conflict as follows:

```
<<<<<<< lao
||||||| tzu
They both may be called deep and profound.
Deeper and more profound,
The door of all subtleties!
=======

   -- The Way of Lao-Tzu, tr. Wing-tsit Chan
>>>>>>> tao
```

The '-E' or '--show-overlap' option outputs less information than
the '-A' or '--show-all' option, because it outputs only unmerged
changes, and it never outputs the contents of the second file. Thus
the '-E' option acts like the '-e' option, except that it brackets the
first and third files from three-way overlapping changes. Similarly,
'-X' acts like '-x', except it brackets all its (necessarily
overlapping) changes. For example, for the three-way overlapping
change above, the '-E' and '-X' options output the following:

```
<<<<<<< lao
=======

   -- The Way of Lao-Tzu, tr. Wing-tsit Chan
>>>>>>> tao
```

If you are comparing files that have meaningless or uninformative
names, you can use the '-L LABEL' or '--label=LABEL' option to show
alternate names in the '<<<<<<<', '|||||||' and '>>>>>>>' brackets.
This option can be given up to three times, once for each input file.
Thus 'diff3 -A -L X -L Y -L Z A B C' acts like 'diff3 -A A B C', except
that the output looks like it came from files named 'X', 'Y' and 'Z'
rather than from files named 'A', 'B' and 'C'.

1.55   diff.guide/Bypassing ed

```
Generating the Merged Output Directly
=====================================
```

With the '-m' or '--merge' option, 'diff3' outputs the merged file
directly. This is more efficient than using 'ed' to generate it, and

works even with non-text files that 'ed' would reject. If you specify
'-m' without an 'ed' script option, '-A' ('--show-all') is assumed.

   For example, the command 'diff3 -m lao tzu tao' (see
Sample diff3 Input for a copy of the input files) would output the
following:

```
     <<<<<<< tzu
     =======
     The Way that can be told of is not the eternal Way;
     The name that can be named is not the eternal name.
     >>>>>>> tao
     The Nameless is the origin of Heaven and Earth;
     The Named is the mother of all things.
     Therefore let there always be non-being,
       so we may see their subtlety,
     And let there always be being,
       so we may see their result.
     The two are the same,
     But after they are produced,
       they have different names.
     <<<<<<< lao
     ||||||| tzu
     They both may be called deep and profound.
     Deeper and more profound,
     The door of all subtleties!
     =======

       -- The Way of Lao-Tzu, tr. Wing-tsit Chan
     >>>>>>> tao
```

1.56 diff.guide/Merging Incomplete Lines

How 'diff3' Merges Incomplete Lines
===================================

   With '-m', incomplete lines (see Incomplete Lines) are simply copied
to the output as they are found; if the merged output ends in an
conflict and one of the input files ends in an incomplete line,
succeeding '|||||||', '=======' or '>>>>>>>' brackets appear somewhere
other than the start of a line because they are appended to the
incomplete line.

   Without '-m', if an 'ed' script option is specified and an
incomplete line is found, 'diff3' generates a warning and acts as if a
newline had been present.

1.57 diff.guide/Saving the Changed File

Saving the Changed File
=======================

   Traditional Unix 'diff3' generates an 'ed' script without the
trailing 'w' and and 'q' commands that save the changes. System V
'diff3' generates these extra commands. GNU 'diff3' normally behaves
like traditional Unix 'diff3', but with the '-i' option it behaves like
System V 'diff3' and appends the 'w' and 'q' commands.

   The '-i' option requires one of the 'ed' script options '-AeExX3',
and is incompatible with the merged output option '-m'.

1.58   diff.guide/Interactive Merging

Interactive Merging with 'sdiff'
****

   With 'sdiff', you can merge two files interactively based on a
side-by-side '-y' format comparison (see Side by Side). Use '-o FILE'
or '--output=FILE' to specify where to put the merged text. See
Invoking sdiff, for more details on the options to 'sdiff'.

   Another way to merge files interactively is to use the Emacs Lisp
package 'emerge'. See emerge, for more information.

   sdiff Option Summary   Summary of 'sdiff' options.
   Merge Commands   Merging two files interactively.

1.59   diff.guide/sdiff Option Summary

Specifying 'diff' Options to 'sdiff'
====================================

   The following 'sdiff' options have the same meaning as for 'diff'.
See diff Options, for the use of these options.

     -a -b -d -i -t -v
     -B -H -I REGEXP

     --ignore-blank-lines --ignore-case
     --ignore-matching-lines=REGEXP --ignore-space-change
     --left-column --minimal --speed-large-files
     --suppress-common-lines --expand-tabs
     --text --version --width=COLUMNS

   For historical reasons, 'sdiff' has alternate names for some
options. The '-l' option is equivalent to the '--left-column' option,
and similarly '-s' is equivalent to '--suppress-common-lines'. The

meaning of the 'sdiff' '-w' and '-W' options is interchanged from that
of 'diff': with 'sdiff', '-w COLUMNS' is equivalent to
'--width=COLUMNS', and '-W' is equivalent to '--ignore-all-space'.
'sdiff' without the '-o' option is equivalent to 'diff' with the '-y'
or '--side-by-side' option (see Side by Side).

1.60   diff.guide/Merge Commands

Merge Commands
==============

   Groups of common lines, with a blank gutter, are copied from the
first file to the output. After each group of differing lines, 'sdiff'
prompts with '%' and pauses, waiting for one of the following commands.
Follow each command with RET.

'e'
     Discard both versions. Invoke a text editor on an empty temporary
     file, then copy the resulting file to the output.

'eb'
     Concatenate the two versions, edit the result in a temporary file,
     then copy the edited result to the output.

'el'
     Edit a copy of the left version, then copy the result to the
     output.

'er'
     Edit a copy of the right version, then copy the result to the
     output.

'l'
     Copy the left version to the output.

'q'
     Quit.

'r'
     Copy the right version to the output.

's'
     Silently copy common lines.

'v'
     Verbosely copy common lines. This is the default.

   The text editor invoked is specified by the 'EDITOR' environment
variable if it is set. The default is system-dependent.

1.61 diff.guide/Merging with patch

```
Merging with 'patch'
********************
```

   'patch' takes comparison output produced by 'diff' and applies the
differences to a copy of the original file, producing a patched
version. With 'patch', you can distribute just the changes to a set of
files instead of distributing the entire file set; your correspondents
can apply 'patch' to update their copy of the files with your changes.
'patch' automatically determines the diff format, skips any leading or
trailing headers, and uses the headers to determine which file to
patch. This lets your correspondents feed an article or message
containing a difference listing directly to 'patch'.

   'patch' detects and warns about common problems like forward
patches. It saves the original version of the files it patches, and
saves any patches that it could not apply. It can also maintain a
'patchlevel.h' file to ensures that your correspondents apply diffs in
the proper order.

   'patch' accepts a series of diffs in its standard input, usually
separated by headers that specify which file to patch. It applies
'diff' hunks (see Hunks) one by one. If a hunk does not exactly match
the original file, 'patch' uses heuristics to try to patch the file as
well as it can. If no approximate match can be found, 'patch' rejects
the hunk and skips to the next hunk. 'patch' normally replaces each
file F with its new version, saving the original file in 'F.orig', and
putting reject hunks (if any) into 'F.rej'.

   See Invoking patch, for detailed information on the options to
'patch'. See Backups, for more information on how 'patch' names backup
files. See Rejects, for more information on where 'patch' puts reject
hunks.

```
  patch Input              Selecting the type of 'patch' input.
  Imperfect                Dealing with imperfect patches.
  Empty Files              Removing empty files after patching.
  Multiple Patches         Handling multiple patches in a file specially.
  patch Messages           Messages and questions 'patch' can produce.
```

1.62 diff.guide/patch Input

```
Selecting the 'patch' Input Format
==================================
```

   'patch' normally determines which 'diff' format the patch file uses
by examining its contents. For patch files that contain particularly
confusing leading text, you might need to use one of the following
options to force 'patch' to interpret the patch file as a certain
format of diff. The output formats listed here are the only ones that
'patch' can understand.

```
`-c'
`--context'
    context diff.

`-e'
`--ed'
    `ed' script.

`-n'
`--normal'
    normal diff.

`-u'
`--unified'
    unified diff.
```

1.63 diff.guide/Imperfect

```
Applying Imperfect Patches
==========================
```

   `patch' tries to skip any leading text in the patch file, apply the
diff, and then skip any trailing text. Thus you can feed a news article
or mail message directly to `patch', and it should work. If the entire
diff is indented by a constant amount of white space, `patch'
automatically ignores the indentation.

   However, certain other types of imperfect input require user
intervention.

```
  Changed White Space        When tabs and spaces don't match exactly.
  Reversed Patches           Applying reversed patches correctly.
  Inexact                    Helping `patch' find close matches.
```

1.64 diff.guide/Changed White Space

```
Applying Patches with Changed White Space
-----------------------------------------
```

   Sometimes mailers, editors, or other programs change spaces into
tabs, or vice versa. If this happens to a patch file or an input file,
the files might look the same, but `patch' will not be able to match
them properly. If this problem occurs, use the `-l' or
`--ignore-white-space' option, which makes `patch' compare white space
loosely so that any sequence of white space in the patch file matches
any sequence of white space in the input files. Non-white-space
characters must still match exactly. Each line of the context must
still match a line in the input file.

1.65   diff.guide/Reversed Patches

```
Applying Reversed Patches
-------------------------
```

   Sometimes people run 'diff' with the new file first instead of
second. This creates a diff that is "reversed". To apply such
patches, give 'patch' the '-R' or '--reverse' option. 'patch' then
attempts to swap each hunk around before applying it. Rejects come out
in the swapped format. The '-R' option does not work with 'ed' scripts
because there is too little information in them to reconstruct the
reverse operation.

   Often 'patch' can guess that the patch is reversed. If the first
hunk of a patch fails, 'patch' reverses the hunk to see if it can apply
it that way. If it can, 'patch' asks you if you want to have the '-R'
option set; if it can't, 'patch' continues to apply the patch normally.
This method cannot detect a reversed patch if it is a normal diff and
the first command is an append (which should have been a delete) since
appends always succeed, because a null context matches anywhere. But
most patches add or change lines rather than delete them, so most
reversed normal diffs begin with a delete, which fails, and 'patch'
notices.

   If you apply a patch that you have already applied, 'patch' thinks
it is a reversed patch and offers to un-apply the patch. This could be
construed as a feature. If you did this inadvertently and you don't
want to un-apply the patch, just answer 'n' to this offer and to the
subsequent "apply anyway" question--or type 'C-c' to kill the 'patch'
process.

1.66   diff.guide/Inexact

```
Helping 'patch' Find Inexact Matches
------------------------------------
```

   For context diffs, and to a lesser extent normal diffs, 'patch' can
detect when the line numbers mentioned in the patch are incorrect, and
it attempts to find the correct place to apply each hunk of the patch.
As a first guess, it takes the line number mentioned in the hunk, plus
or minus any offset used in applying the previous hunk. If that is not
the correct place, 'patch' scans both forward and backward for a set of
lines matching the context given in the hunk.

   First 'patch' looks for a place where all lines of the context
match. If it cannot find such a place, and it is reading a context or
unified diff, and the maximum fuzz factor is set to 1 or more, then
'patch' makes another scan, ignoring the first and last line of
context. If that fails, and the maximum fuzz factor is set to 2 or

more, it makes another scan, ignoring the first two and last two lines
of context are ignored. It continues similarly if the maximum fuzz
factor is larger.

   The '-F LINES' or '--fuzz=LINES' option sets the maximum fuzz factor
to LINES. This option only applies to context and unified diffs; it
ignores up to LINES lines while looking for the place to install a
hunk. Note that a larger fuzz factor increases the odds of making a
faulty patch. The default fuzz factor is 2; it may not be set to more
than the number of lines of context in the diff, ordinarily 3.

   If 'patch' cannot find a place to install a hunk of the patch, it
writes the hunk out to a reject file (see Rejects, for information on
how reject files are named). It writes out rejected hunks in context
format no matter what form the input patch is in. If the input is a
normal or 'ed' diff, many of the contexts are simply null. The line
numbers on the hunks in the reject file may be different from those in
the patch file: they show the approximate location where 'patch' thinks
the failed hunks belong in the new file rather than in the old one.

   As it completes each hunk, 'patch' tells you whether the hunk
succeeded or failed, and if it failed, on which line (in the new file)
'patch' thinks the hunk should go. If this is different from the line
number specified in the diff, it tells you the offset. A single large
offset *may* indicate that 'patch' installed a hunk in the wrong place.
'patch' also tells you if it used a fuzz factor to make the match, in
which case you should also be slightly suspicious.

   'patch' cannot tell if the line numbers are off in an 'ed' script,
and can only detect wrong line numbers in a normal diff when it finds a
change or delete command. It may have the same problem with a context
diff using a fuzz factor equal to or greater than the number of lines
of context shown in the diff (typically 3). In these cases, you should
probably look at a context diff between your original and patched input
files to see if the changes make sense. Compiling without errors is a
pretty good indication that the patch worked, but not a guarantee.

   'patch' usually produces the correct results, even when it must make
many guesses. However, the results are guaranteed only when the patch
is applied to an exact copy of the file that the patch was generated
from.

1.67   diff.guide/Empty Files

Removing Empty Files
====================

   Sometimes when comparing two directories, the first directory
contains a file that the second directory does not. If you give 'diff'
the '-N' or '--new-file' option, it outputs a diff that deletes the
contents of this file. By default, 'patch' leaves an empty file after
applying such a diff. The '-E' or '--remove-empty-files' option to
'patch' deletes output files that are empty after applying the diff.

1.68   diff.guide/Multiple Patches

```
Multiple Patches in a File
==========================
```

   If the patch file contains more than one patch, 'patch' tries to
apply each of them as if they came from separate patch files. This
means that it determines the name of the file to patch for each patch,
and that it examines the leading text before each patch for file names
and prerequisite revision level (see Making Patches, for more on that
topic).

   For the second and subsequent patches in the patch file, you can give
options and another original file name by separating their argument
lists with a '+'. However, the argument list for a second or
subsequent patch may not specify a new patch file, since that does not
make sense.

   For example, to tell 'patch' to strip the first three slashes from
the name of the first patch in the patch file and none from subsequent
patches, and to use 'code.c' as the first input file, you can use:

```
    patch -p3 code.c + -p0 < patchfile
```

   The '-S' or '--skip' option ignores the current patch from the patch
file, but continue looking for the next patch in the file. Thus, to
ignore the first and third patches in the patch file, you can use:

```
    patch -S + + -S + < patch file
```

1.69   diff.guide/patch Messages

```
Messages and Questions from 'patch'
===================================
```

   'patch' can produce a variety of messages, especially if it has
trouble decoding its input. In a few situations where it's not sure how
to proceed, 'patch' normally prompts you for more information from the
keyboard. There are options to suppress printing non-fatal messages
and stopping for keyboard input.

   The message 'Hmm...' indicates that 'patch' is reading text in the
patch file, attempting to determine whether there is a patch in that
text, and if so, what kind of patch it is.

   You can inhibit all terminal output from 'patch', unless an error
occurs, by using the '-s', '--quiet', or '--silent' option.

   There are two ways you can prevent 'patch' from asking you any

questions. The '-f' or '--force' option assumes that you know what you
are doing. It assumes the following:

   * skip patches that do not contain file names in their headers;

   * patch files even though they have the wrong version for the
     'Prereq:' line in the patch;

   * assume that patches are not reversed even if they look like they
     are.

   The '-t' or '--batch' option is similar to '-f', in that it
suppresses questions, but it makes somewhat different assumptions:

   * skip patches that do not contain file names in their headers (the
     same as '-f');

   * skip patches for which the file has the wrong version for the
     'Prereq:' line in the patch;

   * assume that patches are reversed if they look like they are.

   'patch' exits with a non-zero status if it creates any reject files.
When applying a set of patches in a loop, you should check the exit
status, so you don't apply a later patch to a partially patched file.

1.70   diff.guide/Making Patches

Tips for Making Patch Distributions
****

   Here are some things you should keep in mind if you are going to
distribute patches for updating a software package.

   Make sure you have specified the file names correctly, either in a
context diff header or with an 'Index:' line. If you are patching
files in a subdirectory, be sure to tell the patch user to specify a
'-p' or '--strip' option as needed. Take care to not send out reversed
patches, since these make people wonder whether they have already
applied the patch.

   To save people from partially applying a patch before other patches
that should have gone before it, you can make the first patch in the
patch file update a file with a name like 'patchlevel.h' or
'version.c', which contains a patch level or version number. If the
input file contains the wrong version number, 'patch' will complain
immediately.

   An even clearer way to prevent this problem is to put a 'Prereq:'
line before the patch. If the leading text in the patch file contains a
line that starts with 'Prereq:', 'patch' takes the next word from that
line (normally a version number) and checks whether the next input file
contains that word, preceded and followed by either white space or a
newline. If not, 'patch' prompts you for confirmation before

proceeding. This makes it difficult to accidentally apply patches in
the wrong order.

    Since 'patch' does not handle incomplete lines properly, make sure
that all the source files in your program end with a newline whenever
you release a version.

    To create a patch that changes an older version of a package into a
newer version, first make a copy of the older version in a scratch
directory. Typically you do that by unpacking a 'tar' or 'shar'
archive of the older version.

    You might be able to reduce the size of the patch by renaming or
removing some files before making the patch. If the older version of
the package contains any files that the newer version does not, or if
any files have been renamed between the two versions, make a list of
'rm' and 'mv' commands for the user to execute in the old version
directory before applying the patch. Then run those commands yourself
in the scratch directory.

    If there are any files that you don't need to include in the patch
because they can easily be rebuilt from other files (for example,
'TAGS' and output from 'yacc' and 'makeinfo'), replace the versions in
the scratch directory with the newer versions, using 'rm' and 'ln' or
'cp'.

    Now you can create the patch. The de-facto standard 'diff' format
for patch distributions is context format with two lines of context,
produced by giving 'diff' the '-C 2' option. Do not use less than two
lines of context, because 'patch' typically needs at least two lines
for proper operation. Give 'diff' the '-P' option in case the newer
version of the package contains any files that the older one does not.
Make sure to specify the scratch directory first and the newer
directory second.

    Add to the top of the patch a note telling the user any 'rm' and
'mv' commands to run before applying the patch. Then you can remove
the scratch directory.

1.71   diff.guide/Invoking cmp

Invoking 'cmp'
****

    The 'cmp' command compares two files, and if they differ, tells the
first byte and line number where they differ. Its arguments are as
follows:

    cmp OPTIONS... FROM-FILE [TO-FILE]

    The file name '-' is always the standard input. 'cmp' also uses the
standard input if one file name is omitted.

    An exit status of 0 means no differences were found, 1 means some

differences were found, and 2 means trouble.

  cmp Options   Summary of options to 'cmp'.

1.72   diff.guide/cmp Options

Options to 'cmp'
================

   Below is a summary of all of the options that GNU 'cmp' accepts.
Most options have two equivalent names, one of which is a single letter
preceded by '-', and the other of which is a long name preceded by
'--'. Multiple single letter options (unless they take an argument)
can be combined into a single command line word: '-cl' is equivalent to
'-c -l'.

'-c'
     Print the differing characters. Display control characters as a
     '^' followed by a letter of the alphabet and precede characters
     that have the high bit set with 'M-' (which stands for "meta").

'--ignore-initial=BYTES'
     Ignore any differences in the the first BYTES bytes of the input
     files. Treat files with fewer than BYTES bytes as if they are
     empty.

'-l'
     Print the (decimal) offsets and (octal) values of all differing
     bytes.

'--print-chars'
     Print the differing characters. Display control characters as a
     '^' followed by a letter of the alphabet and precede characters
     that have the high bit set with 'M-' (which stands for "meta").

'--quiet'
'-s'
'--silent'
     Do not print anything; only return an exit status indicating
     whether the files differ.

'--verbose'
     Print the (decimal) offsets and (octal) values of all differing
     bytes.

'-v'
'--version'
     Output the version number of 'cmp'.

1.73   diff.guide/Invoking diff

```
Invoking 'diff'
***************
```

   The format for running the 'diff' command is:

     diff OPTIONS... FROM-FILE TO-FILE

   In the simplest case, 'diff' compares the contents of the two files
FROM-FILE and TO-FILE. A file name of '-' stands for text read from
the standard input. As a special case, 'diff - -' compares a copy of
standard input to itself.

   If FROM-FILE is a directory and TO-FILE is not, 'diff' compares the
file in FROM-FILE whose file name is that of TO-FILE, and vice versa.
The non-directory file must not be '-'.

   If both FROM-FILE and TO-FILE are directories, 'diff' compares
corresponding files in both directories, in alphabetical order; this
comparison is not recursive unless the '-r' or '--recursive' option is
given. 'diff' never compares the actual contents of a directory as if
it were a file. The file that is fully specified may not be standard
input, because standard input is nameless and the notion of "file with
the same name" does not apply.

   'diff' options begin with '-', so normally FROM-FILE and TO-FILE may
not begin with '-'. However, '--' as an argument by itself treats the
remaining arguments as file names even if they begin with '-'.

   An exit status of 0 means no differences were found, 1 means some
differences were found, and 2 means trouble.

  diff Options   Summary of options to 'diff'.

1.74   diff.guide/diff Options

```
Options to 'diff'
=================
```

   Below is a summary of all of the options that GNU 'diff' accepts.
Most options have two equivalent names, one of which is a single letter
preceded by '-', and the other of which is a long name preceded by
'--'. Multiple single letter options (unless they take an argument)
can be combined into a single command line word: '-ac' is equivalent to
'-a -c'. Long named options can be abbreviated to any unique prefix of
their name. Brackets ([and]) indicate that an option takes an
optional argument.

'-LINES'
     Show LINES (an integer) lines of context. This option does not
     specify an output format by itself; it has no effect unless it is

combined with '-c' (see Context Format) or '-u' (see
Unified Format). This option is obsolete. For proper operation,
'patch' typically needs at least two lines of context.

'-a'
    Treat all files as text and compare them line-by-line, even if they
    do not seem to be text. See Binary.

'-b'
    Ignore changes in amount of white space. See White Space.

'-B'
    Ignore changes that just insert or delete blank lines. See
    Blank Lines.

'--binary'
    Read and write data in binary mode. See Binary.

'--brief'
    Report only whether the files differ, not the details of the
    differences. See Brief.

'-c'
    Use the context output format. See Context Format.

'-C LINES'
'--context[=LINES]'
    Use the context output format, showing LINES (an integer) lines of
    context, or three if LINES is not given. See Context Format. For
    proper operation, 'patch' typically needs at least two lines of
    context.

'--changed-group-format=FORMAT'
    Use FORMAT to output a line group containing differing lines from
    both files in if-then-else format. See Line Group Formats.

'-d'
    Change the algorithm perhaps find a smaller set of changes. This
    makes 'diff' slower (sometimes much slower). See diff Performance.

'-D NAME'
    Make merged '#ifdef' format output, conditional on the preprocessor
    macro NAME. See If-then-else.

'-e'
'--ed'
    Make output that is a valid 'ed' script. See ed Scripts.

'--exclude=PATTERN'
    When comparing directories, ignore files and subdirectories whose
    basenames match PATTERN. See Comparing Directories.

'--exclude-from=FILE'
    When comparing directories, ignore files and subdirectories whose
    basenames match any pattern contained in FILE. See
    Comparing Directories.

'--expand-tabs'
     Expand tabs to spaces in the output, to preserve the alignment of
     tabs in the input files. See Tabs.

'-f'
     Make output that looks vaguely like an 'ed' script but has changes
     in the order they appear in the file. See Forward ed.

'-F REGEXP'
     In context and unified format, for each hunk of differences, show
     some of the last preceding line that matches REGEXP. See
     Specified Headings.

'--forward-ed'
     Make output that looks vaguely like an 'ed' script but has changes
     in the order they appear in the file. See Forward ed.

'-h'
     This option currently has no effect; it is present for Unix
     compatibility.

'-H'
     Use heuristics to speed handling of large files that have numerous
     scattered small changes. See diff Performance.

'--horizon-lines=LINES'
     Do not discard the last LINES lines of the common prefix and the
     first LINES lines of the common suffix. See diff Performance.

'-i'
     Ignore changes in case; consider upper- and lower-case letters
     equivalent. See Case Folding.

'-I REGEXP'
     Ignore changes that just insert or delete lines that match REGEXP.
     See Specified Folding.

'--ifdef=NAME'
     Make merged if-then-else output using NAME. See If-then-else.

'--ignore-all-space'
     Ignore white space when comparing lines. See White Space.

'--ignore-blank-lines'
     Ignore changes that just insert or delete blank lines. See
     Blank Lines.

'--ignore-case'
     Ignore changes in case; consider upper- and lower-case to be the
     same. See Case Folding.

'--ignore-matching-lines=REGEXP'
     Ignore changes that just insert or delete lines that match REGEXP.
     See Specified Folding.

'--ignore-space-change'
     Ignore changes in amount of white space. See White Space.

‘--initial-tab’
       Output a tab rather than a space before the text of a line in
       normal or context format. This causes the alignment of tabs in
       the line to look normal. See Tabs.

‘-l’
       Pass the output through ‘pr’ to paginate it. See Pagination.

‘-L LABEL’
       Use LABEL instead of the file name in the context format (see
       Context Format) and unified format (see Unified Format) headers.
       See RCS.

‘--label=LABEL’
       Use LABEL instead of the file name in the context format (see
       Context Format) and unified format (see Unified Format) headers.

‘--left-column’
       Print only the left column of two common lines in side by side
       format. See Side by Side Format.

‘--line-format=FORMAT’
       Use FORMAT to output all input lines in if-then-else format. See
       Line Formats.

‘--minimal’
       Change the algorithm to perhaps find a smaller set of changes.
       This makes ‘diff’ slower (sometimes much slower). See
       diff Performance.

‘-n’
       Output RCS-format diffs; like ‘-f’ except that each command
       specifies the number of lines affected. See RCS.

‘-N’
‘--new-file’
       In directory comparison, if a file is found in only one directory,
       treat it as present but empty in the other directory. See
       Comparing Directories.

‘--new-group-format=FORMAT’
       Use FORMAT to output a group of lines taken from just the second
       file in if-then-else format. See Line Group Formats.

‘--new-line-format=FORMAT’
       Use FORMAT to output a line taken from just the second file in
       if-then-else format. See Line Formats.

‘--old-group-format=FORMAT’
       Use FORMAT to output a group of lines taken from just the first
       file in if-then-else format. See Line Group Formats.

‘--old-line-format=FORMAT’
       Use FORMAT to output a line taken from just the first file in
       if-then-else format. See Line Formats.

'-p'
     Show which C function each change is in. See C Function Headings.

'-P'
     When comparing directories, if a file appears only in the second
     directory of the two, treat it as present but empty in the other.
     See Comparing Directories.

'--paginate'
     Pass the output through 'pr' to paginate it. See Pagination.

'-q'
     Report only whether the files differ, not the details of the
     differences. See Brief.

'-r'
     When comparing directories, recursively compare any subdirectories
     found. See Comparing Directories.

'--rcs'
     Output RCS-format diffs; like '-f' except that each command
     specifies the number of lines affected. See RCS.

'--recursive'
     When comparing directories, recursively compare any subdirectories
     found. See Comparing Directories.

'--report-identical-files'
     Report when two files are the same. See Comparing Directories.

'-s'
     Report when two files are the same. See Comparing Directories.

'-S FILE'
     When comparing directories, start with the file FILE. This is
     used for resuming an aborted comparison. See
     Comparing Directories.

'--sdiff-merge-assist'
     Print extra information to help 'sdiff'. 'sdiff' uses this option
     when it runs 'diff'. This option is not intended for users to use
     directly.

'--show-c-function'
     Show which C function each change is in. See C Function Headings.

'--show-function-line=REGEXP'
     In context and unified format, for each hunk of differences, show
     some of the last preceding line that matches REGEXP. See
     Specified Headings.

'--side-by-side'
     Use the side by side output format. See Side by Side Format.

'--speed-large-files'
     Use heuristics to speed handling of large files that have numerous
     scattered small changes. See diff Performance.

'--starting-file=FILE'
     When comparing directories, start with the file FILE. This is
     used for resuming an aborted comparison. See
     Comparing Directories.

'--suppress-common-lines'
     Do not print common lines in side by side format. See
     Side by Side Format.

'-t'
     Expand tabs to spaces in the output, to preserve the alignment of
     tabs in the input files. See Tabs.

'-T'
     Output a tab rather than a space before the text of a line in
     normal or context format. This causes the alignment of tabs in
     the line to look normal. See Tabs.

'--text'
     Treat all files as text and compare them line-by-line, even if they
     do not appear to be text. See Binary.

'-u'
     Use the unified output format. See Unified Format.

'--unchanged-group-format=FORMAT'
     Use FORMAT to output a group of common lines taken from both files
     in if-then-else format. See Line Group Formats.

'--unchanged-line-format=FORMAT'
     Use FORMAT to output a line common to both files in if-then-else
     format. See Line Formats.

'--unidirectional-new-file'
     When comparing directories, if a file appears only in the second
     directory of the two, treat it as present but empty in the other.
     See Comparing Directories.

'-U LINES'
'--unified[=LINES]'
     Use the unified output format, showing LINES (an integer) lines of
     context, or three if LINES is not given. See Unified Format. For
     proper operation, 'patch' typically needs at least two lines of
     context.

'-v'
'--version'
     Output the version number of 'diff'.

'-w'
     Ignore white space when comparing lines. See White Space.

'-W COLUMNS'
'--width=COLUMNS'
     Use an output width of COLUMNS in side by side format. See
     Side by Side Format.

'-x PATTERN'
     When comparing directories, ignore files and subdirectories whose
     basenames match PATTERN. See Comparing Directories.

'-X FILE'
     When comparing directories, ignore files and subdirectories whose
     basenames match any pattern contained in FILE. See
     Comparing Directories.

'-y'
     Use the side by side output format. See Side by Side Format.

1.75   diff.guide/Invoking diff3

Invoking 'diff3'
****

   The 'diff3' command compares three files and outputs descriptions of
their differences. Its arguments are as follows:

     diff3 OPTIONS... MINE OLDER YOURS

   The files to compare are MINE, OLDER, and YOURS. At most one of
these three file names may be '-', which tells 'diff3' to read the
standard input for that file.

   An exit status of 0 means 'diff3' was successful, 1 means some
conflicts were found, and 2 means trouble.

  diff3 Options   Summary of options to 'diff3'.

1.76   diff.guide/diff3 Options

Options to 'diff3'
==================

   Below is a summary of all of the options that GNU 'diff3' accepts.
Multiple single letter options (unless they take an argument) can be
combined into a single command line argument.

'-a'
     Treat all files as text and compare them line-by-line, even if they
     do not appear to be text. See Binary.

'-A'
     Incorporate all changes from OLDER to YOURS into MINE, surrounding
     all conflicts with bracket lines. See Marking Conflicts.

`‘-e’`
    Generate an `‘ed’` script that incorporates all the changes from
    OLDER to YOURS into MINE. See Which Changes.

`‘-E’`
    Like `‘-e’`, except bracket lines from overlapping changes' first
    and third files. See Marking Conflicts. With `‘-e’`, an
    overlapping change looks like this:

        <<<<<<< MINE
        lines from MINE
        =======
        lines from YOURS
        >>>>>>> YOURS

`‘--ed’`
    Generate an `‘ed’` script that incorporates all the changes from
    OLDER to YOURS into MINE. See Which Changes.

`‘--easy-only’`
    Like `‘-e’`, except output only the nonoverlapping changes. See
    Which Changes.

`‘-i’`
    Generate `‘w’` and `‘q’` commands at the end of the `‘ed’` script for
    System V compatibility. This option must be combined with one of
    the `‘-AeExX3’` options, and may not be combined with `‘-m’`. See
    Saving the Changed File.

`‘--initial-tab’`
    Output a tab rather than two spaces before the text of a line in
    normal format. This causes the alignment of tabs in the line to
    look normal. See Tabs.

`‘-L LABEL’`
`‘--label=LABEL’`
    Use the label LABEL for the brackets output by the `‘-A’`, `‘-E’` and
    `‘-X’` options. This option may be given up to three times, one for
    each input file. The default labels are the names of the input
    files. Thus `‘diff3 -L X -L Y -L Z -m A B C’` acts like `‘diff3 -m A
    B C’`, except that the output looks like it came from files named
    `‘X’`, `‘Y’` and `‘Z’` rather than from files named `‘A’`, `‘B’` and `‘C’`.
    See Marking Conflicts.

`‘-m’`
`‘--merge’`
    Apply the edit script to the first file and send the result to
    standard output. Unlike piping the output from `‘diff3’` to `‘ed’`,
    this works even for binary files and incomplete lines. `‘-A’` is
    assumed if no edit script option is specified. See Bypassing ed.

`‘--overlap-only’`
    Like `‘-e’`, except output only the overlapping changes. See
    Which Changes.

`‘--show-all’`
    Incorporate all unmerged changes from OLDER to YOURS into MINE,

surrounding all overlapping changes with bracket lines. See
Marking Conflicts.

'--show-overlap'
     Like '-e', except bracket lines from overlapping changes' first
     and third files. See Marking Conflicts.

'-T'
     Output a tab rather than two spaces before the text of a line in
     normal format. This causes the alignment of tabs in the line to
     look normal. See Tabs.

'--text'
     Treat all files as text and compare them line-by-line, even if they
     do not appear to be text. See Binary.

'-v'
'--version'
     Output the version number of 'diff3'.

'-x'
     Like '-e', except output only the overlapping changes. See
     Which Changes.

'-X'
     Like '-E', except output only the overlapping changes. In other
     words, like '-x', except bracket changes as in '-E'. See
     Marking Conflicts.

'-3'
     Like '-e', except output only the nonoverlapping changes. See
     Which Changes.

1.77   diff.guide/Invoking patch

Invoking 'patch'
****

   Normally 'patch' is invoked like this:

     patch <PATCHFILE

   The full format for invoking 'patch' is:

     patch OPTIONS... [ORIGFILE [PATCHFILE]] [+ OPTIONS... [ORIGFILE]]...

   If you do not specify PATCHFILE, or if PATCHFILE is '-', 'patch'
reads the patch (that is, the 'diff' output) from the standard input.

   You can specify one or more of the original files as ORIG arguments;
each one and options for interpreting it is separated from the others
with a '+'. See Multiple Patches, for more information.

   If you do not specify an input file on the command line, 'patch'

tries to figure out from the "leading text" (any text in the patch that
comes before the `diff` output) which file to edit. In the header of a
context or unified diff, `patch` looks in lines beginning with `***`,
`---`, or `+++`; among those, it chooses the shortest name of an
existing file. Otherwise, if there is an `Index:` line in the leading
text, `patch` tries to use the file name from that line. If `patch`
cannot figure out the name of an existing file from the leading text,
it prompts you for the name of the file to patch.

   If the input file does not exist or is read-only, and a suitable RCS
or SCCS file exists, `patch` attempts to check out or get the file
before proceeding.

   By default, `patch` replaces the original input file with the
patched version, after renaming the original file into a backup file
(see Backups, for a description of how `patch` names backup files).
You can also specify where to put the output with the `-o OUTPUT-FILE`
or `--output=OUTPUT-FILE` option.

  patch Directories   Changing directory and stripping directories.
  Backups   Backup file names.
  Rejects   Reject file names.
  patch Options   Summary table of options to `patch`.

1.78   diff.guide/patch Directories

Applying Patches in Other Directories
=====================================

   The `-d DIRECTORY` or `--directory=DIRECTORY` option to `patch`
makes directory DIRECTORY the current directory for interpreting both
file names in the patch file, and file names given as arguments to
other options (such as `-B` and `-o`). For example, while in a news
reading program, you can patch a file in the `/usr/src/emacs` directory
directly from the article containing the patch like this:

     | patch -d /usr/src/emacs

   Sometimes the file names given in a patch contain leading
directories, but you keep your files in a directory different from the
one given in the patch. In those cases, you can use the `-p[NUMBER]`
or `--strip[=NUMBER]` option to set the file name strip count to
NUMBER. The strip count tells `patch` how many slashes, along with the
directory names between them, to strip from the front of file names.
`-p` with no NUMBER given is equivalent to `-p0`. By default, `patch`
strips off all leading directories, leaving just the base file names,
except that when a file name given in the patch is a relative file name
and all of its leading directories already exist, `patch` does not
strip off the leading directory. (A "relative" file name is one that
does not start with a slash.)

   `patch` looks for each file (after any slashes have been stripped)
in the current directory, or if you used the `-d DIRECTORY` option, in

that directory.

   For example, suppose the file name in the patch file is
'/gnu/src/emacs/etc/NEWS'. Using '-p' or '-p0' gives the entire file
name unmodified, '-p1' gives 'gnu/src/emacs/etc/NEWS' (no leading
slash), '-p4' gives 'etc/NEWS', and not specifying '-p' at all gives
'NEWS'.

1.79   diff.guide/Backups

Backup File Names
=================

   Normally, 'patch' renames an original input file into a backup file
by appending to its name the extension '.orig', or '~' on systems that
do not support long file names. The '-b BACKUP-SUFFIX' or
'--suffix=BACKUP-SUFFIX' option uses BACKUP-SUFFIX as the backup
extension instead.

   Alternately, you can specify the extension for backup files with the
'SIMPLE_BACKUP_SUFFIX' environment variable, which the options override.

    'patch' can also create numbered backup files the way GNU Emacs
does. With this method, instead of having a single backup of each file,
'patch' makes a new backup file name each time it patches a file. For
example, the backups of a file named 'sink' would be called,
successively, 'sink.~1~', 'sink.~2~', 'sink.~3~', etc.

   The '-V BACKUP-STYLE' or '--version-control=BACKUP-STYLE' option
takes as an argument a method for creating backup file names. You can
alternately control the type of backups that 'patch' makes with the
'VERSION_CONTROL' environment variable, which the '-V' option
overrides. The value of the 'VERSION_CONTROL' environment variable and
the argument to the '-V' option are like the GNU Emacs
'version-control' variable (see emacs, for more information on backup
versions in Emacs). They also recognize synonyms that are more
descriptive. The valid values are listed below; unique abbreviations
are acceptable.

't'
'numbered'
     Always make numbered backups.

'nil'
'existing'
     Make numbered backups of files that already have them, simple
     backups of the others. This is the default.

'never'
'simple'
     Always make simple backups.

   Alternately, you can tell 'patch' to prepend a prefix, such as a
directory name, to produce backup file names. The '-B BACKUP-PREFIX'

or '--prefix=BACKUP-PREFIX' option makes backup files by prepending
BACKUP-PREFIX to them. If you use this option, 'patch' ignores any
'-b' option that you give.

    If the backup file already exists, 'patch' creates a new backup file
name by changing the first lowercase letter in the last component of
the file name into uppercase. If there are no more lowercase letters
in the name, it removes the first character from the name. It repeats
this process until it comes up with a backup file name that does not
already exist.

    If you specify the output file with the '-o' option, that file is
the one that is backed up, not the input file.

1.80   diff.guide/Rejects

Reject File Names
=================

    The names for reject files (files containing patches that 'patch'
could not find a place to apply) are normally the name of the output
file with '.rej' appended (or '#' on systems that do not support long
file names).

    Alternatively, you can tell 'patch' to place all of the rejected
patches in a single file. The '-r REJECT-FILE' or
'--reject-file=REJECT-FILE' option uses REJECT-FILE as the reject file
name.

1.81   diff.guide/patch Options

Options to 'patch'
==================

    Here is a summary of all of the options that 'patch' accepts. Older
versions of 'patch' do not accept long-named options or the '-t', '-E',
or '-V' options.

    Multiple single-letter options that do not take an argument can be
combined into a single command line argument (with only one dash).
Brackets ([and]) indicate that an option takes an optional argument.

'-b BACKUP-SUFFIX'
     Use BACKUP-SUFFIX as the backup extension instead of '.orig' or
     '~'. See Backups.

'-B BACKUP-PREFIX'
     Use BACKUP-PREFIX as a prefix to the backup file name. If this
     option is specified, any '-b' option is ignored. See Backups.

'--batch'
     Do not ask any questions. See patch Messages.

'-c'
'--context'
     Interpret the patch file as a context diff. See patch Input.

'-d DIRECTORY'
'--directory=DIRECTORY'
     Makes directory DIRECTORY the current directory for interpreting
     both file names in the patch file, and file names given as
     arguments to other options. See patch Directories.

'-D NAME'
     Make merged if-then-else output using FORMAT. See If-then-else.

'--debug=NUMBER'
     Set internal debugging flags. Of interest only to 'patch'
     patchers.

'-e'
'--ed'
     Interpret the patch file as an 'ed' script. See patch Input.

'-E'
     Remove output files that are empty after the patches have been
     applied. See Empty Files.

'-f'
     Assume that the user knows exactly what he or she is doing, and do
     not ask any questions. See patch Messages.

'-F LINES'
     Set the maximum fuzz factor to LINES. See Inexact.

'--force'
     Assume that the user knows exactly what he or she is doing, and do
     not ask any questions. See patch Messages.

'--forward'
     Ignore patches that 'patch' thinks are reversed or already applied.
     See also '-R'. See Reversed Patches.

'--fuzz=LINES'
     Set the maximum fuzz factor to LINES. See Inexact.

'--help'
     Print a summary of the options that 'patch' recognizes, then exit.

'--ifdef=NAME'
     Make merged if-then-else output using FORMAT. See If-then-else.

'--ignore-white-space'
'-l'
     Let any sequence of white space in the patch file match any
     sequence of white space in the input file. See
     Changed White Space.

'-n'
'--normal'
     Interpret the patch file as a normal diff. See patch Input.

'-N'
     Ignore patches that 'patch' thinks are reversed or already applied.
     See also '-R'. See Reversed Patches.

'-o OUTPUT-FILE'
'--output=OUTPUT-FILE'
     Use OUTPUT-FILE as the output file name. See patch Options.

'-p[NUMBER]'
     Set the file name strip count to NUMBER. See patch Directories.

'--prefix=BACKUP-PREFIX'
     Use BACKUP-PREFIX as a prefix to the backup file name. If this
     option is specified, any '-b' option is ignored. See Backups.

'--quiet'
     Work silently unless an error occurs. See patch Messages.

'-r REJECT-FILE'
     Use REJECT-FILE as the reject file name. See Rejects.

'-R'
     Assume that this patch was created with the old and new files
     swapped. See Reversed Patches.

'--reject-file=REJECT-FILE'
     Use REJECT-FILE as the reject file name. See Rejects.

'--remove-empty-files'
     Remove output files that are empty after the patches have been
     applied. See Empty Files.

'--reverse'
     Assume that this patch was created with the old and new files
     swapped. See Reversed Patches.

'-s'
     Work silently unless an error occurs. See patch Messages.

'-S'
     Ignore this patch from the patch file, but continue looking for
     the next patch in the file. See Multiple Patches.

'--silent'
     Work silently unless an error occurs. See patch Messages.

'--skip'
     Ignore this patch from the patch file, but continue looking for
     the next patch in the file. See Multiple Patches.

'--strip[=NUMBER]'
     Set the file name strip count to NUMBER. See patch Directories.

`--suffix=BACKUP-SUFFIX'
     Use BACKUP-SUFFIX as the backup extension instead of `.orig' or
     `~'. See Backups.

`-t'
     Do not ask any questions. See patch Messages.

`-u'
`--unified'
     Interpret the patch file as a unified diff. See patch Input.

`-v'
     Output the revision header and patch level of `patch'.

`-V BACKUP-STYLE'
     Select the kind of backups to make. See Backups.

`--version'
     Output the revision header and patch level of `patch', then exit.

`--version=control=BACKUP-STYLE'
     Select the kind of backups to make. See Backups.

`-x NUMBER'
     Set internal debugging flags. Of interest only to `patch'
     patchers.

1.82   diff.guide/Invoking sdiff

Invoking `sdiff'
****

   The `sdiff' command merges two files and interactively outputs the
results. Its arguments are as follows:

     sdiff -o OUTFILE OPTIONS... FROM-FILE TO-FILE

   This merges FROM-FILE with TO-FILE, with output to OUTFILE. If
FROM-FILE is a directory and TO-FILE is not, `sdiff' compares the file
in FROM-FILE whose file name is that of TO-FILE, and vice versa.
FROM-FILE and TO-FILE may not both be directories.

   `sdiff' options begin with `-', so normally FROM-FILE and TO-FILE
may not begin with `-'.  However, `--' as an argument by itself treats
the remaining arguments as file names even if they begin with `-'. You
may not use `-' as an input file.

   An exit status of 0 means no differences were found, 1 means some
differences were found, and 2 means trouble.

   `sdiff' without `-o' (or `--output') produces a side-by-side
difference. This usage is obsolete; use `diff --side-by-side' instead.

```
   sdiff Options                    Summary of options to 'diff'.
```

1.83   diff.guide/sdiff Options

```
Options to 'sdiff'
==================

    Below is a summary of all of the options that GNU 'sdiff' accepts.
Each option has two equivalent names, one of which is a single letter
preceded by '-', and the other of which is a long name preceded by
'--'.  Multiple single letter options (unless they take an argument)
can be combined into a single command line argument.  Long named
options can be abbreviated to any unique prefix of their name.

'-a'
     Treat all files as text and compare them line-by-line, even if they
     do not appear to be text.  See Binary.

'-b'
     Ignore changes in amount of white space.  See White Space.

'-B'
     Ignore changes that just insert or delete blank lines.  See
     Blank Lines.

'-d'
     Change the algorithm to perhaps find a smaller set of changes.
     This makes 'sdiff' slower (sometimes much slower).  See
     diff Performance.

'-H'
     Use heuristics to speed handling of large files that have numerous
     scattered small changes.  See diff Performance.

'--expand-tabs'
     Expand tabs to spaces in the output, to preserve the alignment of
     tabs in the input files.  See Tabs.

'-i'
     Ignore changes in case; consider upper- and lower-case to be the
     same.  See Case Folding.

'-I REGEXP'
     Ignore changes that just insert or delete lines that match REGEXP.
     See Specified Folding.

'--ignore-all-space'
     Ignore white space when comparing lines.  See White Space.

'--ignore-blank-lines'
     Ignore changes that just insert or delete blank lines.  See
     Blank Lines.
```

'--ignore-case'
     Ignore changes in case; consider upper- and lower-case to be the
     same. See Case Folding.

'--ignore-matching-lines=REGEXP'
     Ignore changes that just insert or delete lines that match REGEXP.
     See Specified Folding.

'--ignore-space-change'
     Ignore changes in amount of white space. See White Space.

'-l'
'--left-column'
     Print only the left column of two common lines. See
     Side by Side Format.

'--minimal'
     Change the algorithm to perhaps find a smaller set of changes.
     This makes 'sdiff' slower (sometimes much slower). See
     diff Performance.

'-o FILE'
'--output=FILE'
     Put merged output into FILE. This option is required for merging.

'-s'
'--suppress-common-lines'
     Do not print common lines. See Side by Side Format.

'--speed-large-files'
     Use heuristics to speed handling of large files that have numerous
     scattered small changes. See diff Performance.

'-t'
     Expand tabs to spaces in the output, to preserve the alignment of
     tabs in the input files. See Tabs.

'--text'
     Treat all files as text and compare them line-by-line, even if they
     do not appear to be text. See Binary.

'-v'
'--version'
     Output the version number of 'sdiff'.

'-w COLUMNS'
'--width=COLUMNS'
     Use an output width of COLUMNS. See Side by Side Format. Note
     that for historical reasons, this option is '-W' in 'diff', '-w'
     in 'sdiff'.

'-W'
     Ignore horizontal white space when comparing lines. See
     White Space. Note that for historical reasons, this option is
     '-w' in 'diff', '-W' in 'sdiff'.

1.84   diff.guide/Incomplete Lines

```
Incomplete Lines
****************
```

   When an input file ends in a non-newline character, its last line is
called an "incomplete line" because its last character is not a
newline. All other lines are called "full lines" and end in a newline
character. Incomplete lines do not match full lines unless differences
in white space are ignored (see White Space).

   An incomplete line is normally distinguished on output from a full
line by a following line that starts with '\'. However, the RCS format
(see RCS) outputs the incomplete line as-is, without any trailing
newline or following line. The side by side format normally represents
incomplete lines as-is, but in some cases uses a '\' or '/' gutter
marker; See Side by Side. The if-then-else line format preserves a
line's incompleteness with '%L', and discards the newline with '%l';
See Line Formats. Finally, with the 'ed' and forward 'ed' output
formats (see Output Formats) 'diff' cannot represent an incomplete
line, so it pretends there was a newline and reports an error.

   For example, suppose 'F' and 'G' are one-byte files that contain
just 'f' and 'g', respectively. Then 'diff F G' outputs

```
    1c1
    < f
    \ No newline at end of file
    ---
    > g
    \ No newline at end of file
```

(The exact message may differ in non-English locales.) 'diff -n F G'
outputs the following without a trailing newline:

```
    d1 1
    a1 1
    g
```

   'diff -e F G' reports two errors and outputs the following:

```
    1c
    g
    .
```

1.85   diff.guide/Projects

```
Future Projects
***************
```

   Here are some ideas for improving GNU 'diff' and 'patch'. The GNU
project has identified some improvements as potential programming
projects for volunteers. You can also help by reporting any bugs that

you find.

   If you are a programmer and would like to contribute something to the
GNU project, please consider volunteering for one of these projects. If
you are seriously contemplating work, please write to
`gnu@prep.ai.mit.edu' to coordinate with other volunteers.

1.86 diff.guide/Shortcomings

Suggested Projects for Improving GNU `diff' and `patch'
==

   One should be able to use GNU `diff' to generate a patch from any
pair of directory trees, and given the patch and a copy of one such
tree, use `patch' to generate a faithful copy of the other.
Unfortunately, some changes to directory trees cannot be expressed using
current patch formats; also, `patch' does not handle some of the
existing formats. These shortcomings motivate the following suggested
projects.

1.87 diff.guide/Changing Structure

Handling Changes to the Directory Structure

   `diff' and `patch' do not handle some changes to directory
structure. For example, suppose one directory tree contains a directory
named `D' with some subsidiary files, and another contains a file with
the same name `D'.  `diff -r' does not output enough information for
`patch' to transform the the directory subtree into the file.

   There should be a way to specify that a file has been deleted without
having to include its entire contents in the patch file. There should
also be a way to tell `patch' that a file was renamed, even if there is
no way for `diff' to generate such information.

   These problems can be fixed by extending the `diff' output format to
represent changes in directory structure, and extending `patch' to

understand these extensions.

1.88   diff.guide/Special Files

Files that are Neither Directories Nor Regular Files

   Some files are neither directories nor regular files: they are
unusual files like symbolic links, device special files, named pipes,
and sockets. Currently, 'diff' treats symbolic links like regular
files; it treats other special files like regular files if they are
specified at the top level, but simply reports their presence when
comparing directories. This means that 'patch' cannot represent changes
to such files. For example, if you change which file a symbolic link
points to, 'diff' outputs the difference between the two files, instead
of the change to the symbolic link.

   'diff' should optionally report changes to special files specially,
and 'patch' should be extended to understand these extensions.

1.89   diff.guide/Unusual File Names

File Names that Contain Unusual Characters

   When a file name contains an unusual character like a newline or
white space, 'diff -r' generates a patch that 'patch' cannot parse.
The problem is with format of 'diff' output, not just with 'patch',
because with odd enough file names one can cause 'diff' to generate a
patch that is syntactically correct but patches the wrong files. The
format of 'diff' output should be extended to handle all possible file
names.

1.90   diff.guide/Arbitrary Limits

Arbitrary Limits

   GNU 'diff' can analyze files with arbitrarily long lines and files
that end in incomplete lines. However, 'patch' cannot patch such
files. The 'patch' internal limits on line lengths should be removed,
and 'patch' should be extended to parse 'diff' reports of incomplete
lines.

1.91   diff.guide/Large Files

```
Handling Files that Do Not Fit in Memory
-----------------------------------------
```

   'diff' operates by reading both files into memory. This method
fails if the files are too large, and 'diff' should have a fallback.

   One way to do this is to scan the files sequentially to compute hash
codes of the lines and put the lines in equivalence classes based only
on hash code. Then compare the files normally. This does produce some
false matches.

   Then scan the two files sequentially again, checking each match to
see whether it is real. When a match is not real, mark both the
"matching" lines as changed. Then build an edit script as usual.

   The output routines would have to be changed to scan the files
sequentially looking for the text to print.

1.92   diff.guide/Ignoring Changes

```
Ignoring Certain Changes
------------------------
```

   It would be nice to have a feature for specifying two strings, one in
FROM-FILE and one in TO-FILE, which should be considered to match.
Thus, if the two strings are 'foo' and 'bar', then if two lines differ
only in that 'foo' in file 1 corresponds to 'bar' in file 2, the lines
are treated as identical.

   It is not clear how general this feature can or should be, or what
syntax should be used for it.

1.93   diff.guide/Bugs

```
Reporting Bugs
==============
```

   If you think you have found a bug in GNU 'cmp', 'diff', 'diff3',
'sdiff', or 'patch', please report it by electronic mail to
'bug-gnu-utils@prep.ai.mit.edu'. Send as precise a description of the
problem as you can, including sample input files that produce the bug,
if applicable.

   Because Larry Wall has not released a new version of 'patch' since
mid 1988 and the GNU version of 'patch' has been changed since then,
please send bug reports for 'patch' by electronic mail to both
'bug-gnu-utils@prep.ai.mit.edu' and 'lwall@netlabs.com'.

1.94   diff.guide/Concept Index

```
Concept Index
*************
```