GNU wdiff, version 0.6.4

A word difference finder (and others)
Edition 0.6.4, March 2010

François Pinard
Martin von Gagern

This file documents the `wdiff` command, which compares two files, finding which words have been deleted or added to the first for getting the second. It also documents some other `diff` related tools.

Table of Contents

1 Overview

`wdiff` is a front end to `diff` for comparing files on a word per word basis. It works by creating two temporary files, one word per line, and then executes `diff` on these files. It collects the `diff` output and uses it to produce a nicer display of word differences between the original files.

`mdiff` studies one or more input files altogether, and discovers blocks of items which repeat at more than one place. Items may be lines, words, or units defined by user. When in word mode, `mdiff` compares two files, finding which words have been deleted or added to the first in order to create the second, which is useful when two texts differ only by a few words and paragraphs have been refilled. The program has many output formats and interacts well with terminals and pagers (notably with `less`).

`unify` is able to convert context diffs to unidiff format, or the other way around. Some people just prefer one format and despise the other, it is a religious issue. This program brings peace back to Earth.

`wdiff2` is intended as a replacement to `wdiff`. It aims at supporting the same set of options, but uses `mdiff` instead of `diff` as its backend.

`wdiff`, `mdiff` and `wdiff2` were written by François Pinard, while `unify` has been contributed by Wayne Davison. Please report bugs to `wdiff-bugs@gnu.org`. Include the version number, which you can find by running the program with '`--version`'. Please include in your message sufficient input to reproduce what you got, the output you indeed expected, and careful explanations about the nature of the problem.

2 The word difference finder

There are actually two programs for comparing files on a word per word basis. `wdiff` is a front-end to `diff` as found in the GNU diffutils package. It is quite mature. Its planned successor, `wdiff2`, is a front end to `mdiff`, and as experimental as `mdiff` itself. See Chapter 6 [Experimental], page 18.

A word is anything between whitespace. This is useful for comparing two texts in which a few words have been changed and for which paragraphs have been refilled.

2.1 Invoking `wdiff`

The programs `wdiff` and `wdiff2` aim at providing the same set of command line options. They are described below. See Section 5.2 [wdiff Compatibility], page 16, for a list of differences.

```
wdiff option ... old_file new_file
wdiff option ... -d [diff_file]
```

`wdiff` compares files *old_file* and *new_file* and produces an annotated copy of *new_file* on standard output. The empty string or the string '`-`' denotes standard input, but standard input cannot be used twice in the same invocation. The complete path of a file should be given, a directory name is not accepted. `wdiff` will exit with a status of 0 if no differences were found, a status of 1 if any differences were found, or a status of 2 for any error.

In this documentation, *deleted text* refers to text in *old_file* which is not in *new_file*, while *inserted text* refers to text on *new_file* which is not in *old_file*.

`wdiff` supports the following command line options:

'`--help`'
'`-h`' Print an informative help message describing the options.

'`--version`'
'`-v`' Print the version number of `wdiff` on the standard error output.

'`--no-deleted`'
'`-1`' Avoid producing deleted words on the output. If neither '`-1`' or '`-2`' is selected, the original right margin may be exceeded for some lines.

'`--no-inserted`'
'`-2`' Avoid producing inserted words on the output. When this flag is given, the whitespace in the output is taken from *old_file* instead of *new_file*. If neither '`-1`' or '`-2`' is selected, the original right margin may be exceeded for some lines.

'`--no-common`'
'`-3`' Avoid producing common words on the output. When this option is not selected, common words and whitespace are taken from *new_file*, unless option '`-2`' is given, in which case common words and whitespace are rather taken from *old_file*. When selected, differences are separated from one another by lines of dashes. Moreover, if this option is selected at the same time as '`-1`' or '`-2`', then none of the output will have any emphasis, i.e. no bold or underlining. Finally, if this option is not selected, but both '`-1`' and '`-2`' are, then sections of common words between differences are segregated by lines of dashes.

‘`--ignore-case`’
‘`-i`’ Do not consider case difference while comparing words. Each lower case letter is seen as identical to its upper case equivalent for the purpose of deciding if two words are the same.

‘`--statistics`’
‘`-s`’ On completion, for each file, the total number of words, the number of common words between the files, the number of words deleted or inserted and the number of words that have changed is output. (A changed word is one that has been replaced or is part of a replacement.) Except for the total number of words, all of the numbers are followed by a percentage relative to the total number of words in the file.

‘`--auto-pager`’
‘`-a`’ Some initiatives which were previously automatically taken in previous versions of `wdiff` are now put under the control of this option. By using it, a pager is interposed whenever the `wdiff` output is directed to the user’s terminal. Without this option, no pager will be called, the user is then responsible for explicitly piping `wdiff` output into a pager, if required.

The pager is selected by the value of the *PAGER* environment variable when `wdiff` is run. If *PAGER* is not defined at run time, then a default pager, selected at installation time, will be used instead. A defined but empty value of *PAGER* means no pager at all.

When a pager is interposed through the use of this option, one of the options ‘`-l`’ or ‘`-t`’ is also selected, depending on whether the string ‘`less`’ appears in the pager’s name or not.

It is often useful to define ‘`wdiff`’ as an alias for ‘`wdiff -a`’. However, this *hides* the normal `wdiff` behaviour. The default behaviour may be restored simply by piping the output from `wdiff` through `cat`. This dissociates the output from the user’s terminal.

‘`--printer`’
‘`-p`’ Use over-striking to emphasize parts of the output. Each character of the deleted text is underlined by writing an underscore ‘`_`’ first, then a backspace and then the letter to be underlined. Each character of the inserted text is emboldened by writing it twice, with a backspace in between. This option is not selected by default.

‘`--less-mode`’
‘`-l`’ Use over-striking to emphasize parts of output. This option works as option ‘`-p`’, but also over-strikes whitespace associated with inserted text. `less` shows such whitespace using reverse video. This option is not selected by default. However, it is automatically turned on whenever `wdiff` launches the pager `less`. See option ‘`-a`’.

This option is commonly used in conjunction with `less`:

```
wdiff -l old_file new_file | less
```

‘`--terminal`’
‘`-t`’ Force the production of `termcap` strings for emphasising parts of output, even if the standard output is not associated with a terminal. The `TERM` environment variable must contain the name of a valid `termcap` entry. If the terminal description permits, underlining is used for marking deleted text, while bold or reverse video is used for marking inserted text. This option is not selected by default. However, it is automatically turned on whenever `wdiff` launches a pager, and it is known that the pager is *not* `less`. See option ‘`-a`’.

This option is commonly used when `wdiff` output is not redirected, but sent directly to the user terminal, as in:

```
wdiff -t old_file new_file
```

A common kludge uses `wdiff` together with the pager `more`, as in:

```
wdiff -t old_file new_file | more
```

However, some versions of `more` use `termcap` emphasis for their own purposes, so strange interactions are possible.

‘`--start-delete` *argument*’
‘`-w` *argument*’
Use *argument* as the *start delete* string. This string will be output prior to any sequence of deleted text, to mark where it starts. By default, no start delete string is used unless there is no other means of distinguishing where such text starts; in this case the default start delete string is ‘`[-`’.

‘`--end-delete` *argument*’
‘`-x` *argument*’
Use *argument* as the *end delete* string. This string will be output after any sequence of deleted text, to mark where it ends. By default, no end delete string is used unless there is no other means of distinguishing where such text ends; in this case the default end delete string is ‘`-]`’.

‘`--start-insert` *argument*’
‘`-y` *argument*’
Use *argument* as the *start insert* string. This string will be output prior to any sequence of inserted text, to mark where it starts. By default, no start insert string is used unless there is no other means of distinguishing where such text starts; in this case the default start insert string is ‘`{+`’.

‘`--end-insert` *argument*’
‘`-z` *argument*’
Use *argument* as the *end insert* string. This string will be output after any sequence of inserted text, to mark where it ends. By default, no end insert string is used unless there is no other means of distinguishing where such text ends; in this case the default end insert string is ‘`+}`’.

‘`--avoid-wraps`’
‘`-n`’ Avoid spanning the end of line while showing deleted or inserted text. Any single fragment of deleted or inserted text spanning many lines will be considered as being made up of many smaller fragments not containing a newline. So deleted text, for example, will have an end delete string at the end of each line,

just before the new line, and a start delete string at the beginning of the next line. A long paragraph of inserted text will have each line bracketed between start insert and end insert strings. This behaviour is not selected by default.

'`--diff-input`'
'`-d`' Use single unified diff as input. If no input file is specified, standard input is used instead. This can be used to post-process diffs generated form other applications, like version control systems:

```
svn diff | wdiff -d
```

Note that options '`-p`', '`-t`', and '`-[wxyz]`' are not mutually exclusive. If you use a combination of them, you will merely accumulate the effect of each. Option '`-l`' is a variant of option '`-p`'.

2.2 Actual examples of `wdiff` usage

This section presents a few examples of usage, most of them have been contributed by `wdiff` users.

- Change bars example.
  - This example comes from a discussion with Joe Wells.

    The following command produces a copy of *new_file*, shifted right one space to accommodate change bars since the last revision, ignoring those changes coming only from paragraph refilling. Any line with new or changed text will get a '`|`' in column 1. However, deleted text is not shown nor marked.

    ```
    wdiff -1n old_file new_file |
      sed -e 's/^/  /;/{+/s/^ /|/;s/{+//g;s/+}//g'
    ```

    Here is how it works. Word differences are found, paying attention only to additions, as requested by option '`-1`'. For bigger changes which span line boundaries, the insert bracket strings are repeated on each output line, as requested by option '`-n`'. This output is then reformatted with a `sed` script which shifts the text right two columns, turns the initial space into a bar only if there is some new text on that line, then removes all insert bracket strings.
- LaTeX example.
  -

    This example has been provided by Steve Fisk.

    The following uses LaTeX to put deleted text in boxes, and new text in double boxes:

    ```
    wdiff -w "\fbox{" -x "}" -y "\fbox{\fbox{" -z "}}" ...
    ```

    works nicely.
- `troff` example.
  - This example comes from Paul Fox.

    Using `wdiff`, with some `troff`-specific delimiters gives *much* better output. The delimiters I used:

    ```
    wdiff -w'\s-5' -x'\s0' -y'\fB' -z'\fP' ...
    ```

This makes the pointsize of deletions 5 points smaller than normal, and emboldens insertions. Fantastic!

I experimented with:

```
wdiff -w'\fI' -x'\fP' -y'\fB' -z'\fP'
```

since that's more like the defaults you use for terminals or printers, but since I actually use italics for emphasis in my documents, I thought the point size thing was clearer.

I tried it on code, and it works surprisingly well there, too...

- Marty Leisner says:

  In the previous example, you had smaller text being taken out and bold face inserted. I had smaller text being taken out and larger text being inserted, I'm using bold face for other things, so this is more clear.

  ```
  wdiff -w '\s-3' -x'\s0' -y'\s+3' -z'\s0'
  ```

- Colored output example.
  - This example comes from Martin von Gagern.

    If you like colored output, and your terminal supports ANSI escape sequences, you can use this invocation:

    ```
    wdiff -n \
      -w $'\033[30;41m' -x $'\033[0m' \
      -y $'\033[30;42m' -z $'\033[0m' \
      ... | less -R
    ```

    This will print deleted text black on red, and inserted text black on green, assuming that your normal terminal colors are white on black. Of course you can choose different colors if you prefer.

    The '`$'...''` notation is supported by GNU bash, and maybe other shells as well. If your shell doesn't support it, you might need some more tricks to generate these escape sequences as command line arguments.

On a related note, GNU Emacs users might notice that the interactive function `compare-windows` ignores changes in whitespace, if it is given a numeric argument. If the variable `compare-ignore-case` is non-`nil`, it ignores differences in case as well. So, in a way, this offers a kind of incremental version of `wdiff`.

3 The multi-difference finder

The name `mdiff` stands for *multi*-`diff`, and has the purpose of encompassing the functionnality of a few other `diff`-type programs. The prefix *multi-* also stands for the fact the program is often able to study more than two input files at once.

The theory of operation is simple. The program splits all input files into a sequence of items, which may be lines or words. `mdiff` is then said to operate either in *line mode* or in *word mode*. It then tries to find sequences of items which are repeated in the input files. Such common sequences are called *clusters* of items, and each occurrence of a repetition is called a cluster *member*. What remains, once all cluster members are conceptually removed from all input files, is a set of *differences*. The role of `mdiff` is to conveniently list either cluster members and differences.

When input files are very similar, it is likely that clusters will encompass many items (lines or words) and differences will be small. So, most listing options inhibit the printing of cluster members. However, one may ask for the few beginning or ending items of cluster members to be printed nevertheless, as a way to provide a kind of feedback or *context* of the difference, those context items are sometimes said to be at the *horizon* of the difference. In merged listings, cluster members may just not be printed, except maybe for a few context items at the beginning of the member (just after a difference), and a few context items at the end of the member (just before a difference).

When cluster members are short, or if you prefer, when the differences are not far away from each other, it is quite possible that the required context items often cover the full extent of the cluster members, which then are not inhibited anymore when this happens. A run of differences intermixed with such non-suppressed members is called a *hunk*. Some reports produced by `mdiff` are showned as a list of hunks, and it is to be understood that common items are elided between hunks. However, each hunk in itself has no item missing, and each item of the hunk is analysed as pertaining either to only one of the input file or to many of them. Each hunk is preceded by a header, which explains the line position of all input files prior to the hunk itself. By comparing a hunk header with the previous hunk header, the user can have a hint about how much printing was spared.

When two input files are quite similar, clusters are usually presented in the same order in all files. If a cluster member A in the first file corresponds to a cluster member A in the second file, it is likely that another cluster member B which appears *after* A in the first file will correspond to a cluster member B in the second file which appears *after* A as well. So, in many cases, while producing merged listing of files, cluster members may be made to naturally correspond to one another. However, this is not always true, in particular when the second file has been produced from the first by moving a big chunk of code away from its original position. In such cases, we say that members have *crossed*. When members are crossed and `mdiff` has to make a merged listing, it selects one cluster member as being *naturally* associated with its correspondant (either the pair of A's or the pair of B's) and then consider the other cluster as being part of a difference. The crossed nature of the member may still be analysed and reported, or it may be ignored.

The standard `diff` program is meant for when there are exactly two input files, for which crossed members should be ignored. `mdiff` output format has been designed in such a way that it should resemble `diff` output for this precise case. However, `diff` formats are not sufficient for representing all cases which `mdiff` may address, and this is not mature

yet. That is why `mdiff`, in its current state, still experiments with output formats, which are subject to change.

When the input files are not very similar, or rather different, merged listings are not very significant nor useful, and may even be rather confusing. The best to do in such cases is using `mdiff` for making an annotated relisting of all input files, in which cluster members are properly identified and referred to one another.

Statistics.

```
Read summary: 137 files, 41975 lines
Work summary: 439 clusters, 1608 members, 8837 duplicate lines
```

The summary lines, triggered by the '`-s`' option, say that about 8837 non-ignorable lines could be removed over the 41975 which has been read, by using functions, `#include`, `#define`, or similar devices.

If one manages to execute `mdiff` within GNU Emacs so the output described above is collected into the `*compilation*` buffer, the command *C-`* ('`M-x next-error`') will proceed to the next cluster member in the other window, and similarily for other compilation mode commands. This is a useful way for handling `mdiff` output.

Each line in the hunk, after the header, comes from the compared files, but is shifted right so the first column (or the first few columns) of each line gives information about where the line is coming from. A space indicates a line which is common to all files. In case there are only two input files, a minus sign indicates a line from the first file and a plus sign a line from the second file. Else, a letter from '`a`' to '`z`', or more than one letter if there are more than 26 files, indicates to which file the line pertains. If a line or a block of line pertains to many files but not to all of them, the first column holds a vertical bar, and the line or block of lines is bracketed between '`@/`' and '`@\`' lines, which are kind of comments within the hunk. The initial bracket lists all file letters that are related to the incoming line.

I initially wrote `mdiff` specifically to help cleaning a C++ project which was a bit large, and in which many big monolithic classes were derived from each other most probably by rough copying followed by local modifications. I intended to fragment most common clusters and segregate the parts into virtual methods in outer classes, and override these methods, as appropriate, with less common variants within inner classes. `mdiff` was good at pointing me to exactly where I should look at. Of course, it never did the cleanup for me, but it helped doing the research about what should be done. Reusing `mdiff` over the half-cleaned project gave me more fine grained analysis of what was left to consider.

3.1 Invoking `mdiff`

The format for running the `mdiff` program is:

```
mdiff option ... file ...
```

`mdiff` read all input *file*s and produces its results on standard output. Optionally, standard error might receive a progress report or a few statistics.

`wdiff` compares files *old_file* and *new_file* and produces an annotated copy of *new_file* on standard output. The empty string or the string '`-`' denotes standard input, but standard input cannot be used twice in the same invocation. The complete path of a file should be

given, a directory name is not accepted. `wdiff` will exit with a status of 0 if no differences were found, a status of 1 if any differences were found, or a status of 2 for any error.

In this documentation, *deleted text* refers to text in *old_file* which is not in *new_file*, while *inserted text* refers to text on *new_file* which is not in *old_file*.

`mdiff` supports the following command line options:

'`--version`'
: Merely prints the version numbers on standard output, and exits without doing anything else.

'`--help`'
: Merely prints a page of help on standard output, and exits without doing anything else.

'`--threshold=`*number*'
'`-t` *number*'
: Specifies the minimum number of non-ignorable lines which are required for two runs of lines to compare as equal. No cluster member may ever have less than *number* lines. By default, clusters have 4 lines or more.

'`--no-deleted`'
'`-1`'
: Avoid producing deleted words on the output. If neither '`-1`' or '`-2`' is selected, the original right margin may be exceeded for some lines.

'`--no-inserted`'
'`-2`'
: Avoid producing inserted words on the output. When this flag is given, the whitespace in the output is taken from *old_file* instead of *new_file*. If neither '`-1`' or '`-2`' is selected, the original right margin may be exceeded for some lines.

'`--no-common`'
'`-3`'
: Avoid producing common words on the output. When this option is not selected, common words and whitespace are taken from *new_file*, unless option '`-2`' is given, in which case common words and whitespace are rather taken from *old_file*. When selected, differences are separated from one another by lines of dashes. Moreover, if this option is selected at the same time as '`-1`' or '`-2`', then none of the output will have any emphasis, i.e. no bold or underlining. Finally, if this option is not selected, but both '`-1`' and '`-2`' are, then sections of common words between differences are segregated by lines of dashes.

'`--ignore-case`'
'`-i`'
: Do not consider case difference while comparing words. Each lower case letter is seen as identical to its upper case equivalent for the purpose of deciding if two words are the same.

'`--auto-pager`'
'`-A`'
: Some initiatives which were previously automatically taken in previous versions of `wdiff` are now put under the control of this option. By using it, a pager is interposed whenever the `wdiff` output is directed to the user's terminal. Without this option, no pager will be called, the user is then responsible for explicitly piping `wdiff` output into a pager, if required.

  The pager is selected by the value of the *PAGER* environment variable when `wdiff` is run. If *PAGER* is not defined at run time, then a default pager,

selected at installation time, will be used instead. A defined but empty value of *PAGER* means no pager at all.

When a pager is interposed through the use of this option, one of the options '`-l`' or '`-t`' is also selected, depending on whether the string '`less`' appears in the pager's name or not.

It is often useful to define '`wdiff`' as an alias for '`wdiff -a`'. However, this *hides* the normal `wdiff` behaviour. The default behaviour may be restored simply by piping the output from `wdiff` through `cat`. This dissociates the output from the user's terminal.

'`--printer`'
'`-p`' Use over-striking to emphasize parts of the output. Each character of the deleted text is underlined by writing an underscore '`_`' first, then a backspace and then the letter to be underlined. Each character of the inserted text is emboldened by writing it twice, with a backspace in between. This option is not selected by default.

'`--less-mode`'
'`-l`' Use over-striking to emphasize parts of output. This option works as option '`-p`', but also over-strikes whitespace associated with inserted text. `less` shows such whitespace using reverse video. This option is not selected by default. However, it is automatically turned on whenever `wdiff` launches the pager `less`. See option '`-a`'.

This option is commonly used in conjunction with `less`:

```
wdiff -l old_file new_file | less
```

'`--terminal`'
'`-t`' Force the production of `termcap` strings for emphasising parts of output, even if the standard output is not associated with a terminal. The '`TERM`' environment variable must contain the name of a valid `termcap` entry. If the terminal description permits, underlining is used for marking deleted text, while bold or reverse video is used for marking inserted text. This option is not selected by default. However, it is automatically turned on whenever `wdiff` launches a pager, and it is known that the pager is *not* `less`. See option '`-a`'.

This option is commonly used when `wdiff` output is not redirected, but sent directly to the user terminal, as in:

```
wdiff -t old_file new_file
```

A common kludge uses `wdiff` together with the pager `more`, as in:

```
wdiff -t old_file new_file | more
```

However, some versions of `more` use `termcap` emphasis for their own purposes, so strange interactions are possible.

'`--start-delete` *argument*'
'`-w` *argument*'
Use *argument* as the *start delete* string. This string will be output prior to any sequence of deleted text, to mark where it starts. By default, no start delete string is used unless there is no other means of distinguishing where such text starts; in this case the default start delete string is '`[-`'.

‘`--end-delete` *argument*’
‘`-x` *argument*’
: Use *argument* as the *end delete* string. This string will be output after any sequence of deleted text, to mark where it ends. By default, no end delete string is used unless there is no other means of distinguishing where such text ends; in this case the default end delete string is ‘`-]`’.

‘`--start-insert` *argument*’
‘`-y` *argument*’
: Use *argument* as the *start insert* string. This string will be output prior to any sequence of inserted text, to mark where it starts. By default, no start insert string is used unless there is no other means of distinguishing where such text starts; in this case the default start insert string is ‘`{+`’.

‘`--end-insert` *argument*’
‘`-z` *argument*’
: Use *argument* as the *end insert* string. This string will be output after any sequence of inserted text, to mark where it ends. By default, no end insert string is used unless there is no other means of distinguishing where such text ends; in this case the default end insert string is ‘`+}`’.

‘`--avoid-wraps`’
‘`-n`’
: Avoid spanning the end of line while showing deleted or inserted text. Any single fragment of deleted or inserted text spanning many lines will be considered as being made up of many smaller fragments not containing a newline. So deleted text, for example, will have an end delete string at the end of each line, just before the new line, and a start delete string at the beginning of the next line. A long paragraph of inserted text will have each line bracketed between start insert and end insert strings. This behaviour is not selected by default.

Some choices are hard-wired into the program, but might well become options in later releases. For example:

- No cluster may span a file boundary, that is, start near the end of one input file and continue at the beginning of the next file.
- A cluster may have many members from the same file.
- White space is ignored between the beginning of a line and the first non-white character.
- White space is significant when embedded in a line, or when ending a line.
- Lines having no significant part (only white lines for now) are *ignorable*. Such ignorable lines are logically considered as not being part of the input files for the sake of comparisons.
- Comments from the C language are not especially ignored. Unless ignored for other reasons (being white lines), they are indeed significant lines.
- No cluster member may ever directly start nor end with ignorable lines. However, ignorable lines may still be embedded within a cluster member.
- In the generated output, clusters containing the biggest number of ignorable lines are output first, while smaller clusters appear last. All lines pertaining to a single cluster are output together. Within a cluster, members are listed in the order of the initial reading of input files.

Note that options '`-p`', '`-t`', and '`-[wxyz]`' are not mutually exclusive. If you use a combination of them, you will merely accumulate the effect of each. Option '`-l`' is a variant of option '`-p`'.

3.2 Resource considerations and efficiency

Memory consumption

`mdiff` can easily handle medium-sized project. For a 32 bits architecture, the memory requirements may computed like this:

- 8 bytes per file
- 8 bytes per line
- 4 bytes per cluster
- 8 bytes per cluster member

Time consumption

To evaluate the speed, consider the example shown above (see Chapter 3 [mdiff], page 7), and yielding these statistics:

```
Read summary: 137 files, 41975 lines
Work summary: 439 clusters, 1608 members ...
```

Once many files in the memory cache, and redirecting the output to '`/dev/null`', the processing takes 3 seconds of real time on an Intel 486/100, which looks good. I was indeed afraid of some hidden O(n^2) behaviour[1], even if the program is mostly O(n*log(n)). Maybe one will discover or construct cases putting `mdiff` on its knees. So far, `mdiff` seemingly behaves well for the little problems given to it. If we devise and generate a more traditional `diff`-like output, in which all input files are relisted, this will add some time to the processing, but it will be only linear with regard with the total length of input files.

There is a clever optimized sorting algorithm for *all* substrings of a file, which might be generalised to handle words or lines for `mdiff`. But since the program is already faster than we initially expected, there is no emergency to resort to using such an algorithm.

Trading complexity for clarity

When lines repeat a lot, there are surprisingly many ways to relate blocks of lines, and reporting them *all* can make very hairy listings. Any choice about reporting similarities, or not, is somewhat arbitrary, but we ought to make some of such choices for the program to be practical. Some of these choices are detailed here.

If all members of a given cluster A are proper subsets of all members of another given cluster B, then cluster A is wholly forgotten. However, let's presume for example that there are more members in A than in B. Then, some members of A necessarily appear unrelated to any member of B. In such case, it has been decided more useful to report *all* occurrences of A members, even those embedded within occurrences of B members. When only interested in members

[1] n is the total number of lines.

B, annotations pertaining to A may be perceived as clutter. However, when interested in members of A, getting all of them is probably the most useful choice.

It sometimes happen that members of a very same cluster overlap. In the string '`a a a`', there are two overlapping members for the cluster represented by the string '`a a`', one from the first two '`a`', another from the last two '`a`'. In such cases, one member of such an overlap is automatically chopped so the overlap does not occur.

White lines and items containing only delimiters are the possible source of a lot of complexity, if these are fully taken as significant. Since this does not add much to clarity, they are better ignored, usually, through using '`--ignore-blank-lines`' ('`-B`') or '`--ignore-delimiters`' ('`-j`'). Increasing the value of '`--minimum-size=`*`items`*' ('`-J` *`items`*') option also cut off complexity in favor of clarity, yet some small matches may then go unnoticed. Exactly how to best adjust the *items* value is left for the user to decide.

4 The diff format converter

The program `unify` has the purpose of manipulating context diffs and unified context diffs. `unify` will accept either a regular context diff (old- or new-style) or a unified context diff as input, and generate either a unified diff or a new-style context diff as output.

Various other options allow you to echo the non-diff (comment) lines to stderr, modify the diff by removing the comment lines, and/or tweak the diff into a format that is good for releasing patches.

I think most people prefer unified context diffs in general. But some of us just have trouble reading unidiffs, unless they get very simple. Usual context diffs show how the code was *before*, and then, how the code is *after*. Some people just prefer understanding twice thoroughly, than once fuzzily. The tool is useful for those who handle a lot of diffs from various sources, and want them in a uniform format.

4.1 Invoking `unify`

The format for running the `unify` program is:

```
unify option ... [file]
```

The program reads the diff to convert from *file*, or if the source file is not mentioned, it will be read from the standard input. The default is to output the diff in the opposite style of whatever was input, that is, regular context diffs will become unified context diffs, and unified context diffs will become unified context diffs, but this can be overridden by options.

`unify` supports the following command line options:

‘`--version`’
: Merely prints the version numbers on standard output, and exits without doing anything else.

‘`--help`’
: Merely prints a page of help on standard output, and exits without doing anything else.

‘`--context-diffs`’
‘`-c`’
: Forces context diff output.

‘`--echo-comments`’
‘`-e`’
: Echoes non-diff (comment) lines to stderr. If a comment line is being stripped via the ‘`-p`’ option, it is echoed with a preceding ‘`!!!` ’. If all comments are being stripped (via the ‘`-s`’ option), no special designation is given.

‘`--old-diffs`’
‘`-o`’
: Is used to force a context diff to be interpreted as being of the old-style even if it has the extra trailing asterisks that normally mark the new-style. This is only needed if `unify` fails to work with your version of `diff`.

‘`--patch-format`’
‘`-p`’
: Turns on patch-output mode. This will do two things:

  1. Transform a header like:

```
*** orig/file Sat May  5 02:59:37 1990
--- ./file Sat May  5 03:00:08 1990
```

into a line of '`Index: file`' — we choose the shorter name and strip a leading '`./`' sequence if present.

2. Strip lines that begin with '`Only in `', '`Common subdir`', '`Binary files`' or '`diff -`'.

'`-P`' Is the same as '`-p`'.

'`--strip-comments`'
'`-s`' Strips non-diff lines (comments).

'`--unidiffs`'
'`-u`' Forces unified diff output.

'`-U`' Is the same as '`-up`'.

'`--use-equals`'
'`-=`' Will use a '`=`' prefix in a unified diff for lines that are common to both files instead of using a leading space. Though this is harder to read, it is less likely to be mangled by trailing-space-stripping sites when posted to Usenet.

5 How mdiff differs

The GNU project already has a diff program which is part of the GNU diffutils package. There also are various non-GNU diff programs provided by various constructors.

There is also the well-established wdiff which uses diff under the hood. It differs slightly from wdiff2, its intended mdiff-based successor.

The following sections compare mdiff specifications with both GNU diff and with wdiff.

5.1 Differences with diff

GNU diff is a program which matured for a long while, and for which algorithms are based on computer science literature. It is a fast program. By comparison, mdiff is not more than a program kludged up rapidly to satisfy a few precise needs. It only tries not being inordinately slow.

Most diff options are accepted by mdiff under the same short and long option names, and is able to produce resembling output, for making mdiff easier to learn and less surprising to users. Yet, some differences exist in option decoding and output format. Since diff and mdiff use different matching algorithms, it is very likely that the differences will not be exactly analyzed identically.

- A few diff options, which either accept no argument or require a mandatory one, are implemented in mdiff as options accepting an optional argument. This may yield some surprises, for example, '-c4bir' would be accepted by diff and rejected by mdiff, yet it may rewritten '-birc4' for both. See below.
- Options '-c' and '-u' in diff ask for regular context and unified context output, respectively, without specifying the number of lines in the context. diff has '-C *number*' and '-U *number*' options for asking for regular or unified context diffs with *number* context lines. If '-c4' asks for four lines of context, the '4' is not really an argument of '-c', and this is really interpreted as '-c -4', where '-*number*' is meant to be a deprecated option for choosing the number of context lines, option which mdiff does not implement. In mdiff, '-c' and '-u' are really two options which are allowed to receive an optional argument, so the number of lines may, or may not be given, at the choice of the user. In mdiff, options '-C' and '-U' are completely equivalent to '-c' and '-u', and are provided only for the sake of compatibility.
- Option '-v' in diff means '--version', while it means '--verbose' in mdiff. There is no short form for '--version' in mdiff.

5.2 Differences with wdiff

Even if mdiff is meant to fully support wdiff, options have been shuffled around so mdiff could better merge both diff and wdiff options in a common scheme. diff habits were almost always favored in this option reorganisation.

wdiff2 is now a mere front-end to mdiff that only rewrites the options. The following notes apply.

- Some options are just transmitted unchanged, these are '-1', '-2', '-3' and '-i'.
- Option '-c' also gets turned into '-i', to be compatible with wdiff versions up to '0.4'.

- Simple option '`-a`' in `wdiff` becomes '`-A`' in `mdiff`, '`-l`' becomes '`-k`', '`-n`' becomes '`-m`', '`-p`' becomes '`-o`', '`-s`' becomes '`-v`' and '`-t`' becomes '`-z`'.
- Options introducing strings, which are '`-w`', '`-x`', '`-y`' and '`-z`' in `wdiff`, respectively become '`-Y`', '`-Z`', '`-Q`' and '`-R`' in `mdiff`.
- Options '`-C`', '`-h`' and '`-v`' are processed directly by `wdiff` and are not transmitted to `mdiff`.
- Further, the '`-C`' option of `wdiff` has no equivalent in `mdiff`.
- A new option '`-q`' inhibits the message which explains how `mdiff` might have been directly called.
- The option '`--diff-input`' ('`-d`') from `wdiff` isn't supported by `wdiff2` (yet).

6 Experimental programs

The GNU wdiff source package contains sources for a number of tools besides `wdiff` itself. These are considered experimental: they might work for you, but they might just as well fail. The following programs are considered experimental:

- `mdiff`
- `wdiff2`
- `unify`

Building these applications can be configured at build time by passing '`--with-experimental`' to the '`configure`' script.

For this build, they have been enabled. If you encounter a bug in an experimental program, the maintainers would still like to learn about it, but there is a greater chance that they decide not to fix such issues unless you provide a patch as well.

6.1 History of the Experimental programs

Many users suggested features, which were in turn inviting for the integration of `wdiff` into GNU diffutils. Collaboration proved to be rather difficult. After a few years, the `wdiff` author finally gave in and created `mdiff` as a way to break out of the situation and for becoming able to proceed with users' suggestions.

Before `mdiff` and the new `wdiff2` based on it were officially released, the original author resigned maintainership. The new maintainers had little experience with the code, and therefore decided to mark it experimental. That way, the code wouldn't be lost, but it would be clear that it wasn't as testes as the good old `wdiff` command.