fswatch

fswatch

Cross-platform file change monitor with multiple backends
for `fswatch` version 1.6.0, 30 September 2015

Enrico M. Crisostomo

This manual is for `fswatch` (version 1.6.0, 30 September 2015), a cross-platform file change monitor with multiple backends, including Apple OS X File System Events API, *BSD kqueue, Linux inotify, Microsoft Windows and a `stat`-based backend.

Short Contents

Table of Contents

1 History

1.1 fswatch 1.6.0

- `fswatch` can be build on Microsoft Windows using Cygwin.
- A monitor for Microsoft Windows has been added.
- `fswatch` can survive monitor overflows and notify them as a specially crafted change events (of type `Overflow`) if invoked with the `--allow-overflow` option.

1.2 fswatch 1.5.1

- `fswatch-run` scripts have been removed.
- As a consequence of the `fswatch-run` removal, dependency on at least one supported shell (Zsh and Bash) have been removed from the currently supported ports:
  - Macports.
  - Homebrew.
  - FreeBSD.

1.3 fswatch 1.5.0

- Added the `--event` option to allow filtering by event type.

1.4 fswatch 1.4.7

- Fix bug in exclusion filter ordering (PR 75).
- `README.md` improvements.
- Documentation improvements.

1.5 fswatch 1.4.6

- Fix issue 74: Assertion failed on fsw_destroy_session.

1.6 fswatch 1.4.5.3

- Fix issue 67: 100% CPU usage while using libfswatch. This issue only affects the inotify monitor, available only on Linux.

1.7 fswatch 1.4.5.2

- Fix issue 66: Exclude items with poll_monitor not considered.

1.8 fswatch 1.4.5.1

- Do not distribute wrapper scripts for shells which are not installed (the FreeBSD port system checks shebangs and complains).

1.9 fswatch 1.4.5

- Add custom record formats.

1.10 fswatch 1.4.4

- Localize `fswatch` and `libfswatch` using GNU `gettext`.
- Add Italian (`it`) localization.
- Add Spanish (`es`) localization.

1.11 fswatch 1.4.3.2

- Fix `Makefile.am` because of broken link when *DESTDIR* installs are performed.

1.12 fswatch 1.4.3.1

- Fix bug in `fswatch-run` wrapper script for Zsh which caused last argument not to be split when passed to `xargs`.

1.13 fswatch 1.4.3

- Add batch marker feature to delimit the boundaries of a batch of events.
- Add Texinfo documentation.
- `libfswatch` API is now versioned.
- Improved Autoconf checks.
- The inotify monitor now waits for events and honours the latency settings.
- Automaticaly generate the ChangeLog using Git.
- Update `autogen.sh` to honour some commonly used environment variables.

1.14 fswatch 1.4.2

- The inotify monitor now provides the same functionality provided by all the other monitors. Recursive directory monitoring is now implemented.
- Version and revision is now determined dynamically from Git by ancillary scripts invoked by the GNU Build System.

1.15 fswatch 1.4.1.1

- `fswatch` does not compile on OS X < 10.9 because some required C++11 classes are not supported by the C++ runtime.

1.16 fswatch 1.4.1

- `fswatch` does not compile on OS X < 10.9 because some required C++11 classes are not supported by the C++ runtime.

1.17 fswatch 1.4.0

- The `libfswatch` library has been added with bindings for C and C++.
- `fswatch` let users specify the monitor to use by name.

1.18 fswatch 1.3.9

- Fix Issue 23: Add `--include` option.
- Fix Issue 25: Add `--incluse` option.
- Paths can be included using `-i`/`--include` and providing a set of regular expressions.

1.19 fswatch 1.3.8

- Fix Issue 34: Diagnostic messages were output by the inotify monitor even if `fswatch` was not run in verbose mode.

1.20 fswatch 1.3.7

- Fix Issue 32: Problems building `fswatch` 1.3.6 on Mac v. 10.8.5.
- Remove usages of C++11 initializer lists so that fswatch builds with older compiler.

1.21 fswatch 1.3.6

- Fix Issue 26: `fswatch-run` can't run a command with arguments.
- `fswatch-run` scripts are provided for Zsh and Bash.
- System is scanned during installation to check for Zsh and Bash availability. Path of found shells is substituted in the corresponding scripts, otherwise the default `/bin/`*`shell`* is used.
- If a supported shell is found, the `fswatch-run` symbolic link is created in the installation directory to the corresponding script. The lookup order of the shells is:
    - Zsh.
    - Bash.

1.22 fswatch 1.3.5

- Fix Issue 27: Redirect usage text to standard error unless `-h` or `--help`.
- Fix bug to write usage to standard error when invalid arguments are specified.

1.23 fswatch 1.3.4

- Fix bug in `fswatch-run` script to allow arguments to be passed to the command to run.

1.24 fswatch 1.3.3

- Add `-o/--one-per-batch` option to print a single message with the number of change events in the current batch.
- Add `fswatch-run` shell script to mimic the behaviour of earlier `fswatch` versions and launch the specified command when change events are received.

1.25 fswatch 1.3.2

- `fswatch` has been merged with `fsw`.

2 Introduction

`fswatch` is a file change monitor that receives notifications when the contents of the specified files or directories are modified. `fswatch` implements four kinds of monitors:

- A monitor based on the File System Events API of Apple OS X.
- A monitor based on kqueue, an event notification interface introduced in FreeBSD 4.1 and supported on most *BSD systems (including OS X).
- A monitor based on inotify, a Linux kernel subsystem that reports file system changes to applications.
- A monitor based on the Microsoft Windows' `ReadDirectoryChangesW` function and reads change events asynchronously.
- A monitor which periodically stats the file system, saves file modification times in memory and manually calculates file system changes, which can work on any operating system where stat (2) can be used.

`fswatch` should build and work correctly on any system shipping either of the aforementioned APIs.

2.1 History and `fswatch` Authors

Alan Dipert wrote the first implementation of `fswatch` in 2009. This version ran exclusively on OS X and relied on the *FSEvents* API to get change events from the OS.

At the end of 2013 Enrico M. Crisostomo wrote `fsw` aiming at providing not only a drop-in replacement for `fswatch`, but also a common front-end from multiple file system change events APIs, including:

- OS X FSEvents.
- *BSD kqueue.
- Linux inotify.
- Microsoft Windows.

In April 2014 Alan and Enrico, in the best interest of users of either `fswatch` and `fsw`, agreed on merging the two programs together. At the same time, Enrico was taking over `fswatch` as a maintainer.

As a consequence, development of `fswatch` will continue on its main repository while the `fsw` repository will likely be frozen and its documentation updated to redirect users to `fswatch`.

2.2 Reporting Bugs and Suggestions

If you find problems or have suggestions about this program or this manual, please report them as new issues in the official GitHub repository of `fswatch` at https://github.com/emcrisostomo/fswatch. If you wish, you may contact the authors at the addresses listed in the `AUTHORS` file.

When reporting a bug, please be sure to include as much detail as possible.

3 Tutorial Introduction to `fswatch`

This chapter is a tutorial walk-through on the most common use cases where `fswatch` is useful:

- Detecting file system changes.
- Observing file system changes.
- Processing `fswatch` output.

3.1 Detecting File System Changes

A common use case is *detecting* file system changes in a set of file system objects[1] where *details* of a change are irrelevant. This mode of operation is called *bulk mode* and `fswatch` will only dump a single event record per batch[2] containing the number of affected file system objects. No other details are avaible in the event record.

The most common application of this mode of operation is performing a bulk action on all the observed file system objects, such as a synchronization with `rsync`, which will serve us as an example. In this case, the detection of a change triggers the execution of a synchronization script, no matter its kind nor the specific object the event affects.

To run `fswatch` in batch mode, the (`-o`, `--one-per-batch`) must be used:

```
$ fswatch -o path ...
2
10
```

The (`-l`, `--latency`) option can be used to set the latency according to the requirements:

```
$ fswatch -o -l 5 path ...
4
7
```

This way, you can respond to change events in a way which is (or can easily be) path-independent (because you are not receiving any event detail) and you prefer to 'bubble' events together to reduce the overhead of the command being executed.

In bulk mode the output of `fswatch` is guaranteed to have the following structure:

```
number\n
```

where '`number`' is an integer value and '`\n`' is the new line character. A line with this structure is very easy to read with either `xargs` or the `read` builtin:

---

[1] In the context of this manual (unless specified otherwise), *file system object* refers undistinctively to *files* and *directories*.

[2] A *batch* is an iteration of `fswatch` scanning logic, whose frequency is $\nu = l^{-1}$, where l is the *latency*.

```
$ fswatch -o path | while read num ; \
  do \
    ... \
  done
```

In many scripts of this kind, the *num* variable can even be ignored.

3.2 Observing File System Changes

Besides the batch mode, `fswatch` provides a *main* mode providing full details of the kind of events detected and the file system objects they refer to. The main mode is `fswatch`'s default mode of operation and needs no specific flags to be activated.

In this mode, `fswatch` outputs change events to the standard output. By default, only the affected file name is printed and the change event record structure has the following structure:

```
/full/path/to/changed/object\n
```

However, many options are available to format the event record, including:

- The possibility of adding the event timestamp.
- The possibility of adding the event mask in both textual and numerical form.
- The possibility of customizing the event record using a `printf`-like format string.

Since a Unix file name may contain any character but the path separator '`/`' and '`NUL`'[3] the choice of using '`\n`' as record separator may lead to unexpected results (since a file name can legally contain '`\n`'). For this reason, along the line of what other tools such as `find` and `xargs` already do, the '`NUL`' character ('`\0`') can alternatively be used:

```
/full/path/to/changed/object\0
```

3.2.1 Event details

Beside the full path of the change object, details on the kind of change event can be obtained using the (`-x`, `--event-flags`) option:

```
$ fswatch -xr /path/to/observe
/path/to/observe Created Renamed OwnerModified IsFile
...
```

In this case, a space-separated list of change flags are printed after the path of the changed object. The record structure is thus:

```
/full/path/to/changed/object flag ([ ][flag])*
```

[3] Depending on the file system being used, other restrictions may apply. However, for file system portability reasons, you should consider '`NUL`' as the only forbidden character.

where '`flag`' is an event flag. At least one event flag is always present, and additional ones are 'bubbled' into the same record and separated by space. For more information on event flags see [Event Flags], page 19.

3.2.2 Parseability Issues

Since a file name may contain spaces, this record structure is not unambigually parseable if more than one event flag is present: in this case, any subset $[0, x], x < n - 1$ of the n event flags may be part or the file name and hence any parse result would be indeterminate. This issue can be solved using a custom record format (see [Custom Record Formats], page 15).

3.2.3 Numeric Event Flags

Instead of using user-friendly event flag names, as seen in the previous section, *numeric* event flags can be used instead. Currently, the real advantage this method offers, despite possibly cleaner flag-decoding logic, is the availability of a non-ambigous event record representation.

To instruct `fswatch` to print numeric event flags, the (`-n`, `--numeric`) option must be used:

```
$ fswatch -xnr /path/to/observe
/path/to/observe 2058
```

The numeric event flag value is the bitwise OR of the individual event flag values, that are powers of 2. In the previous example, the flag 2058 is decomposed in powers of 2 as $2058 = 2048 + 8 + 2 = 2^{11} + 2^3 + 2$, that is, the eleventh, the third and the first event flags.

3.3 Processing fswatch Output

Very often you wish to not only receive an event, but react to it. The simplest way to do it is piping fswatch output to another process. Since in Unix and Unix-like file system file names may potentially contain any character but '`NUL`' ('`\0`') and the path separator ('`/`'), `fswatch` has a specific mode of operation when its output must be piped to another process. When the (`-0`, `--print0`) option is used, `fswatch` will use the '`NUL`' character as record separator, thus allowing any other character to appear in a path. This is important because many commands and shell builtins (such as `read`) split words and lines by default using the characters in `$IFS`, which by default contains characters which may be present (although rarely) in a file name, resulting in a wrong event path being received and processed.

The simplest way to pipe `fswatch` to another program is using `xargs`:

```
$ fswatch -0 [opts] [paths] | xargs -0 -n 1 -I {} command
```

The command in this example does the following:

- `fswatch -0` will split records using the '`NUL`' character.
- `xargs -0` will split records using the '`NUL`' character. This is required to correctly match impedance with `fswatch`.

- `xargs -n 1` will invoke *`command`* every record. If you want to do it every *x* records, then use `xargs -n` *x*.
- `xargs -I {}` will substitute occurrences of `{}` in command with the parsed argument. If the command you are running does not need the event path name, just delete this option. If you prefer using another replacement string, substitute `{}` with your choice.

3.4 Detecting the Boundaries of a Batch of Changes

If a process or script is piped to **`fswatch`** output, sometimes it would be desirable to detect the 'boundaries' of a batch of changes. This way, the process receiving the stream of changes would rely on the timings imposed by the latency settings of **`fswatch`** to start a phase of events *processing* after a phase or events *gathering*. The **`--batch-marker`** option can be used to accomplish this task:

```
$ fswatch --batch-marker -r ~
/Users/enricomariacrisostomo/.zsh_history.LOCK
NoOp
/Users/enricomariacrisostomo/.zsh_history.new
/Users/enricomariacrisostomo/.zsh_history
/Users/enricomariacrisostomo/.zsh_history.LOCK
NoOp
```

In this example, the '`NoOp`' records mark the end of the 1 second batches of events output by **`fswatch`**. The batch marker can be customized. For more information [Batch Marker], page 17.

3.5 Receiving a Single Event

Another requested feature is the possibility of receiving a single event and exit **`fswatch`**. This is most useful when existing scripts processing events include the restart logic of **`fswatch`**. This use case is implemented by the `-1`, **`--one-event`** option:

```
$ fswatch -1 /path/to/watch
/path/to/watch/child0
/path/to/watch/child1
...
$
```

4 Invoking fswatch

This chapter is about how fswatch is invoked. There are many options and two styles for writing them.

4.1 Synopsis of fswatch

fswatch is invoked with:

```
fswatch [options] [paths]
```

fswatch interprets file names as being relative to the working directory and canonicalizes them using realpath.

If a directory is used as an argument, the directory object is watched and, optionally and depending on the monitor being used, the directory is scanned recursively and all its children are watched as well.

Depending on the monitor being used, recursively scanning huge directory hierarchies or big set of files may be resource consuming, CPU intensive or even impossible. The characteristics of the available monitors in a system should be assessed in order to choose the best monitor according to the specific needs.

Besides successful exits[1], indicated with the exit code 0, fswatch may exit with an error. fswatch will try to print a diagnostic description on stderr when an unexpected error occurs.

The documented[2] exit codes of fswatch are the following:

0 FSW_EXIT_OK: No error occurred.

1 FSW_EXIT_UNK_OPT: An unknown option was input.

2 FSW_EXIT_USAGE: Help message was requested.

3 FSW_EXIT_LATENCY: Invalid latency.

4 FSW_EXIT_STREAM: A stream related problem occurred.

5 FSW_EXIT_ERROR: An unkown error occurred.

6 FSW_EXIT_ENFILE: A file could not be opened.

7 FSW_EXIT_OPT: Unused.

8 FSW_EXIT_MONITOR_NAME: The specified monitor does not exist.

9 FSW_EXIT_FORMAT: The specified monitor is invalid.

---

[1] Depending on the monitor and options being used, fswatch may not exit unless *stopped* with a signal such as TERM or QUIT.

[2] Exit codes are documented in c/error.h of libfswatch.

4.2 The Two Option Styles

`fswatch` implements two option styles which are common in Unix and Unix-like operating systems and GNU software: *short* and *long* options. The biggest difference between short and long options are argument placing (for options taking one).

Whether long options are available in a system depend on the availability of the `getopt_long` function at build time. For this reason, users should familiarise themselves with short options and use them when possible and do not rely on long options to be available on any `fswatch` installation.

4.2.1 Long Options

In systems where `getopt_long` is available, each short option has a corresponding long option with a *mnemonic* name starting with two dashes (e.g.: `--version`).

Long options are meant to be easy to remember and to provide hints about what a command is going to perform. The following command:

```
$ fswatch --event-flags --numeric --recursive ~
```

is clearer than:

```
$ fswatch -xnr ~
```

If a long option takes an argument, it can be specified in two ways, depending on whether the argument is optional or mandatory:

- Separating the argument from the option name with an equal sign, if the argument is of either kind.

  ```
  $ fswatch --latency=5 ~
  ```

- Separating the argument from the option name with any amount of white space, if the argument is mandatory.

  ```
  $ fswatch --latency 5 ~
  ```

4.2.2 Short Options

Most options have a *short* form consisting of a dash followed by a single character, such as `-l` (which is equivalent to `--latency`). When available, a short form is interchangeable with the long one.

If a short option takes an argument, it can be specified in two ways:

- Separating the argument from the option name with any amount of white space:

  ```
  $ fswatch -l 5 ~
  ```

- Joining the argument to the option name:

  ```
  $ fswatch -l5 ~
  ```

Short options can be stuck together provided all the options but the last one take no argument, in which case it can be specified as described above. The command:

```
$ fswatch -xnrl 5 ~
```

is equivalent to:

```
$ fswatch -x -n -r -l 5 ~
```

where '5' is the argument of `-l`.

4.3 fswatch Options

In the following table you can find the list, in alphabetical order, of `fswatch`'s options.

`--print0`
`-0`

Use the ASCII '`NUL`' ('`\0`') as record separator.

`--one-event`
`-1`

Exit `fswatch` after the first set of events is received.

`--allow-overflow`
Sets the allow overflow flag of the monitor. When this flag is set, monitor buffer overflows are reported as change events of type `fsw_event_flag::Overflow`.

`--batch-marker`
Print a marker at the end of every batch.

`--exclude`
`-e`

Exclude paths matching *regex*.

`--extended`
`-E`

Use extended regular expressions.

`--format`

Use the specified record format.

`--format-time`
`-f`

Print the event time using the specified *format*.

`--help`
`-h`

Show the help message.

`--include`
`-i`

Include paths matching *regex*.

`--insensitive`
`-I`

Use case insensitive regular expressions.

`--latency`
`-l`

Set the latency using the specified *value*.

`--follow-links`
`-L`

Follow symbolic links.

`--list-monitors`
`-M`

List the available monitors.

`--monitor`
`-m`

Use the specified *monitor*.

`--numeric`
`-n`

Print a numeric event mask.

`--one-per-batch`
`-o`

Print a single message with the number of change events in the current batch.

`--recursive`
`-r`

Recurse subdirectories.

`--timestamp`
`-t`

Print the event timestamp.

`--utc-time`
`-u`

Print the event time as UTC time.

`--verbose`
`-v`

Print verbose output.

`--version`

Print the version of fswatch and exit.

`--event-flags`
`-x`

Print the event flags.

`--event-flag-separator`
Use the specified string as event flag separator.

4.4 Whitespace and Record Format

As seen in [Observing File System Changes], page 8, file names may contain characters such as '`\n`' which are commonly used as line separators. Many commonly used Unix commands and shell builtins use characters in the `$IFS` environment variable[3] as *separators* to split words. By default, `$IFS` contains the characters ' ' (*SPC*), '`\t`', '`\n`' and '`\0`' ('`NUL`').

Therefore, if a file contains such a separator character (and all but '`NUL`' are *legal*), then a parsing ambiguity may arise when using certain record formats such as:

path[*flag*]+

In this case, for example, if $n > 1$ *flags* are present in the record, and hence more than one ' ' (*SPC*) is present, then it is not known whether any subset containing a number x of consecutive flags ($x < n$) is part of the path or not.

The same reasoning applies when splitting *lines* instead of *words*: since '`\n`' may be a legal file name character, then it is now known whether '`\n`' indicates a record's end or simply is part of a file name.

For this reason, in order to avoid parsing ambiguity, this options instructs `fswatch` to use ASCII '`NUL`' as record separator.

> **Warning:** The use of the `--print0` solves the *line* splitting ambiguity but not the *word* splitting ambiguity when using textual event flags. A solution to this problem is provided by *custom record formats* (see [Custom Record Formats], page 15).
>
> Another way to get an unambiguous record format is using *numeric* event flags (see [Numeric Event Flags], page 21).

4.5 Custom Record Formats

To solve the problem of line splitting ambiguities and to provide users the possibilities of tailoring the record format to their needs, `fswatch` allows users to specify the event record *format* using the `--format` option.

This options requires a `printf`-like[4] *format string* ordinary text containing zero or more *directives*. Characters not belonging to a format directive are copied unchanged to the output, while directives are interpreted and replaced with the result of their evaluation.

---

[3] IFS (Internal Field Separators).

[4] Although the available directive are much less than what `printf` offers.

4.5.1 Format Directives

Directives start with '%' which is always treated as a special character: either it marks the beginning of a directive or it is interpreted as an escape character[5].

The available directives are:

`%%` Inserts the '%' character.

`%0` Inserts an ASCII 'NUL' ('0') character.

`%n` Inserts a *newline* character.

`%f` Inserts the list of event flags, separated by default by the space character (' ') or by the separator specified with the `--event-flag-separator` option (see [Event Flag Separator], page 16).

`%p` Inserts the path.

`%t` Inserts the timestamp, formatted with `strftime` using the format optionally specified with the `--format-time` option.

4.5.2 Record Termination

Each record is terminated by either a newline character ('`\n`') or an ASCII '`NUL`' character when `-0` is specified. The record termination character has the following characteristics:

- It is *not* part of the format string.
- Its value can only be chosen between '`\n`' and '`NUL`' ('`\0`').
- It cannot be suppressed.

4.5.3 Event Flag Separator

When the list of event flags is printed, textual items are separated by default by spaces (' '). The user can specify an alternate event flag separator using the `--event-flag-separator` and passing the desired separator string as argument.

For instance, if the user wants event flags to be separated by a comma, the following command can be used:

```
$ fswatch --event-flag-separator=, -x [OPTIONS] [PATHS]
```

4.5.4 Builtin Formats

The format used by `fswatch` when a custom format is not specified is determined as follows[6]:

- '`%t` ' is added at the beginning of the format string if `-t` is used.
- '`%p`' is always appended to the format string.
- ' `%f`' is added at the end of the format string if `-x` is used.

[5] Which is the same as considering escaped characters the result of a directive.

[6] In the following example, the record termination character is not shown.

4.6 Batch Marker

Since `fswatch` typically outputs an *endless* event stream, processing parties parsing its output may be interested in 'batch event processing': that is, processing batches of events instead of endlessly processing events one by one.

To support this use case, `fswatch` provides the `--batch-marker` option; when specified, `fswatch` will output a customizable 'batch marker record' processing parties can use as batch *delimiters*. Batch demarcation is made naturally using the monitor's processing loop and its latency setting: every time the monitor loops (typically when latency is elapsed), then a batch marker is printed as final record, as shown in the next example:

```
$ fswatch --batch-marker -r ~
/Users/enricomariacrisostomo/.zsh_history.LOCK
NoOp
/Users/enricomariacrisostomo/.zsh_history.new
/Users/enricomariacrisostomo/.zsh_history
/Users/enricomariacrisostomo/.zsh_history.LOCK
NoOp
```

By default, the batch marker takes the form of a single-line record:

```
NoOp[\n | \0]
```

terminated with either '`\n`' or '`NUL`' ('`\0`') depending on other `fswatch` settings. However, the user can customize it by providing the desired marker string as optional argument to `--batch-marker`:

```
% ./fswatch --batch-marker="*** BATCH END ***" -r ~
/Users/enricomariacrisostomo/.zsh_history.LOCK
*** BATCH END ***
```

4.7 Filtering by Path

Filters are *regular expression* which are evaluated against the monitored object path to determine whether a path must be accepted or rejected. Sometimes, the exclusion of a path may result in the exclusion of an object from the list of monitored objects, while other times a path must be evaluated only when an event is detected and in this case the corresponding object cannot be removed from the monitored object list in advance[7].

Event though event *filtering* is commonly performed when processing `fswatch` output, the possibility of filtering paths 'at the source' provides not only a greater amount of flexbility, but also:

- Improved performance, since `fswatch` will only monitor matching objects[8].

---

[7] This behaviour is monitor-specific.

[8] Whether an object whose path is matched by an exclusion filter is monitored or not is a monitor-specific implementation detail.

- Less resource pressure, especially when resource-intensive monitors are used. This is especially important when using monitors that rely on the availability of open file descriptors for any monitored object.
- Simpler processing logic, since part of the path filtering logic is performed by `fswatch`.

Since filters are implemented using the '`regcomp`' library, this feature is built into `fswatch` only on systems where this library is available.

4.7.1 Types of Filters and Order of Execution

Two types of filters are available:

- *Inclusion* filters.
- *Exclusion* filters.

As their name indicates, they are used to include and exclude paths from the monitored object list and from resulting events. `fswatch` processes filters this way:

- If a path matches an including filter, the path is accepted no matter any other filter.
- If a path matches an excluding filter, the path is rejected.
- If a path matches no filters, the path is accepted.

Said another way:

- All paths are accepted *by default*, unless an exclusion filter says otherwise.
- Inclusion filters may override any other exclusion filter.
- The order in the definition of filters in the command line has no effect.

4.7.2 Filter Modifiers

Filters are regular expression executed using the `regcomp` function which is able to interpret case-sensitive and case-insensitive *basic* and *extended* regular expressions as described in Base Definitions volume of IEEE Std 1003.1-2001, Chapter 9, Regular Expressions.

The (`--insensitive`, `-I`) option instructs `fswatch` to use case insensitive regular expressions. The following example adds an exclusion filter so that `fswatch` ignores any file system object whose name ends with *`.log`*, no matter the case.

```
$ fswatch -Ie ".*\.log$" ~
```

The (`--extended`, `-E`) option instructs `fswatch` to use extended regular expressions, such as:

```
$ fswatch -Ee "xl[st]+" ~
```

Treating the characteristics and the difference between different kinds of regular expressions is out of scope in this manual.

4.8 Filtering by Event Type

Events can be filtered by event type passing `fswatch` a list of event type *names* to accept using the `--event` option:

```
$ fswatch -x --event Created --event Removed ~
```

If no event type filters are specified `fswatch` will accept events of any type; on the other hand, as soon as a filter is specified, only events with a matching type will be accepted.

4.9 Latency

The *latency* l, expressed in seconds, is the amount of time that passes between the moment `fswatch` outputs a set of detected changes and the next. What happens during the time in-between is a monitor-specific implementation detail.

Some APIs, such as OS X's FSEvents, implement the concept of latency themselves and `fswatch` appears idle in between. Only when the specified amount of time passes, change events are received, processed and written to standard output. Others, such as Linux's inotify, do not[9]; in this case, the inotify monitor *waits* for events a maximum of l seconds; after that, the monitor logic loops again, performs house-keeping activities[10] and starts waiting again.

The important thing to keep in mind is that latency and a monitor's behaviour are implementation-dependent: check the documentation of the monitor you are using to get further information about how latency is handled.

4.10 Symbolic Links

Symbolic links are commonly used file system objects and, as it is customary for file system utilities, `fswatch` can either *follow* them and monitor the linked object[11] or monitor the link itself.

4.11 Event Flags

Event flags identify the kind of change a file system object has undergone. Many of them directly map to common file system operations (such as creation, deletion, update, etc.), others are less common (such as attribute modification), and others are monitor and platform specific.

Currently, `fswatch` maps monitor-specific event flags to 'global' event flags acting as a sort of 'greatest common denominator' of all the available

[9] inotify publishes changes on a file identified by a descriptor which is `read` by `fswatch`.

[10] Such as re-scanning objects which did not exist in the previous iteration.

[11] When following links, the resolution is recursive: that is, if a link points to another symbolic link, this link is followed as well, and so on, until an object of a different kind is found.

monitor flags. The list of all the available global event flags, defined in `c/cevent.h`, is the following:

`PlatformSpecific`
: This event maps a platform-specific event that has no corresponding flag.

`Created`
: The object has been created.

`Updated`
: The object has been updated. The kind of update is monitor-dependent.

`Removed`
: The object has been removed.

`Renamed`
: The object has been renamed.

`OwnerModified`
: The object's owner has changed.

`AttributeModified`
: An object's attribute has changed.

`MovedFrom`
: The object has moved from this location to a new location of the same file system.

`MovedTo`
: The object has moved from another location in the same file system into this location.

`IsFile`
: The object is a regular file.

`IsDir`
: The object is a directory.

`IsSymLink`
: The object is a symbolic link.

`Link`
: The object link count has changed.

`Overflow`
: The monitor has overflowed.

4.11.1 Peculiarities and Pitfalls

As you can see, the list of event flags contains element whose meaning is overlapping, at least partially. `Link`, for instance, may be equivalent to `Create` or `Removed`, depending on the whether the new link count is 1 or 0. `MovedFrom` and `MovedTo` may be equivalent to `Create` and `Removed` if the monitor is unable to discern a move operation has taken place (which is not always possible, as in the case of the poll monitor).

`fswatch` is unable to univocally map the specific flags of all the monitors consistently. Forcefully, the mapping depends on the capabilities of the monitor which, in turn, depend on the capabilities of the API being used.

For this reason, when processing change events, either the behaviour of the underlying monitor is known and taken into account, or all the flags which could possibly be attached at the operation being looked for must be taken into account.

> **Warning:** As already explained (see [Whitespace and Record Format], page 15), the record format when using event flags in textual form is ambiguous. For this reason, using numeric event flags (see [Numeric Event Flags], page 21) or a custom record format (see [Custom Record Formats], page 15) is recommended when `fswatch` output must be processed.

4.11.2 Numeric Event Flags

When using the (`--numeric`, `-n`) `fswatch` will output event flags in *numeric* format. A change event record may have multiple event flags and the numeric value is calculated as the bitwise `or` of the numeric values of all the flags. Since the value of an event flag is guaranteed to be unique and to be a number n in the form $n = 2^k$ for a certain integer k, then the numeric value of a set of event flags is univocally determined.

To check whether a given event flag is present when processing `fswatch` output, iti s sufficient to check whether its bit is set to 1 in the event value. Let's suppose we want to check whether the event flag whose value is e is present in a record whose flag numerical value is n. If the result r of

$$r = e \wedge b$$

where $\wedge$ is the bitwise `and` operator, is $r > 0$, then the flag e is present in n.

The numeric value of all the event flags is the following:

- `PlatformSpecific`: 1
- `Created`: 2
- `Updated`: 4
- `Removed`: 8
- `Renamed`: 16
- `OwnerModified`: 32
- `AttributeModified`: 64
- `MovedFrom`: 128
- `MovedTo`: 256
- `IsFile`: 512
- `IsDir`: 1024
- `IsSymLink`: 2048
- `Link`: 4096
- `Overflow`: 8192

4.12 Choosing a Monitor

`fswatch` is a front-end to multiple *monitors*, each taking advantage of different monitoring APIs that may be available in a system. When building `fswatch`, `configure` scans the system to check which APIs are available and builds support for all of them.

A 'special' monitor, the *poll* monitor, manually scans the file system looking for differences. This is a fallback monitor for situations where other, more efficient APIs are not available. The poll monitor is available on any system providing the `stat` function.

Although `fswatch` chooses the 'best' monitor between the available ones, a user may wish to use another. A specific monitor can be chosen using the (`--monitor`, `-m`) option. The list of available monitors can be obtained using the (`--list-monitors`, `-M`) option or at the end of the help message:

```
$ fswatch --list-monitors
  fsevents_monitor
  kqueue_monitor
  poll_monitor
$ fswatch --help
[...]
Available monitors in this platform:

  fsevents_monitor
  kqueue_monitor
  poll_monitor
[...]
```

A monitor can then be chosen by passing the mandatory '*`name`*' argument to the `-m` option:

```
$ fswatch -m kqueue_monitor ~
```

In this case, the '`kqueue_monitor`' is manually chosen.

4.13 Recursive Scanning

`fswatch`'s behaviour when scanning a directory may vary on a monitor by monitor basis. The semantics of the (`--recursive`, `-r`) option is: recursively scan subdirectories. However, implementations may silently add '*if the monitor does not do so already*'.

Since each monitor uses a different API, its behaviour depends on that of the backing API, and it is monitor-specific. The OS X FSEvents API, for example, will always recurse subdirectories when monitoring a directory. In this case, even though `-r` is not specified, the monitor will monitor a directory's children nonetheless and there is no way to avoid it[12].

[12] But manually filtering out events based on paths, but `fswatch` does not do so *by design*.

In general, users should always use the `-r` option according to its semantics, no matter what the monitor does. The only case when `-r` is ‘not’ honoured is when a monitor *adds* information by recursively monitoring children even when `-r` is not specified. Please notice that when this happens, there may be no performance overhead since the backing API is specifically designed to behave like this.

The authors think this is not a problem. If you think this behaviour can be improved, please fill a bug report (see [Reporting Bugs and Suggestions], page 5).

4.14 Monitor Tunables

Some monitors may accept monitor-specific parameters to tune their behaviour. At this purpose, `fswatch` offers a mechanism to pass key-value pair which are literally passed to the underlying monitor. A key-value pair `(k, v)` can be passed to a monitor using the `--monitor-property` option:

```
$ fswatch --monitor-property k=v ~
```

Multiple key-value pairs can be passed by using the `--monitor-property` option multiple times.

5 Monitors

`fswatch` is a file system monitoring utility that achieves portability across multiple platform by decoupling the front-end (the `fswatch` itself) from back-end logic. Back-end logic is encapsulated in multiple, system-specific *monitors*, interacting with different monitoring APIs. Since each operating system may ship a different set of APIs[1], each operating system will support the corresponding set of monitors.

The list of available monitors is decided at build time by the `configure` script. Monitors cannot be currently plugged-in but recompiling the `libfswatch` library (shipped with `fswath`). The list of available monitors can be obtained in the help message:

```
$ fswatch --help
[...]
Available monitors in this platform:

  fsevents_monitor
  kqueue_monitor
  poll_monitor
[...]
```

5.1 Available Monitors

Currently, the available monitors are:

- The *FSEvents* monitor, a monitor based on the File System Events API of Apple OS X (see [The FSEvents Monitor], page 26).
- The *kqueue* monitor, a monitor based on *kqueue*, an event notification interface introduced in FreeBSD 4.1 and supported on most *BSD systems (including OS X) (see [The kqueue Monitor], page 26).
- The *inotify* monitor, a Linux kernel subsystem that reports file system changes to applications (see [The inotify Monitor], page 26).
- The *Windows* monitor, a monitor that uses the Microsoft Windows' `ReadDirectoryChangesW` function and reads change events asynchronously.
- The *poll* monitor, a monitor that periodically stats the file system, saves file modification times in memory and manually calculates file system changes, which can work on any operating system where `stat` can be used (see [The Poll Monitor], page 28).

Each monitor has its own strengths, weakness and peculiarities. Although `fswatch` strives to provide a uniform experience no matter which monitor is used, it is still important for users to know which monitor they are using and to be aware of existing bugs, limitations, corner cases or pathological behaviour.

[1] In fact, only OS X supports more than one such API: BSD's *kqueue* and *FSEvents*.

5.2 The FSEvents Monitor

The FSEvents monitor, available only on Apple OS X, has no known limitations and scales very well with the number of files being observed. In fact, I observed no performance degradation when testing `fswatch` observing changes on a filesystem of 500 GB over long periods of time. On OS X, this is the default monitor.

5.2.1 Peculiarities

The (`--recursive`, `-r`) option has no effect when used with the FSEvents monitor since the FSEvents API already monitors a directory's children by default. There is no overhead nor resource-consumption issue with this behaviour, but users processing the output must be aware that for each directory *multiple* events may be generated by its children.

5.3 The kqueue Monitor

The kqueue monitor, available on any *BSD system featuring the `kevent` function, is very similar in principle to other similar APIs (such as FSEvents and inotify) but has important drawback and limitations.

5.3.1 Peculiarities

The kqueue monitor *requires a file descriptor to be opened for every file being watched*. As a result, this monitor scales *badly* with the number of files being observed and may begin to misbehave as soon as the `fswatch` process runs out of file descriptors. In this case, `fswatch` dumps one error on standard error for every file that cannot be opened so that users are notified and can take action, including terminating the `fswatch` session. Beware that on some systems the maximum number of file descriptors that can be opened by a process is set to a *very low value* (values as low as 256 are not uncommon), even if the operating system may allow a much larger value.

If you are running out of file descriptors when using this monitor and you cannot reduce the number of observed items, either:

- Consider raising the number of maximum open file descriptors (check your OS' documentation).
- Consider using another monitor.

5.4 The inotify Monitor

The inotify monitor uses is backed by the inotify API and the `inotify_*` set of functions, introduced on Linux since kernel 2.6.13. Similarly to the FSEvents API, inotify is very efficient, suffers from no known resource-exhaustion problems and scales very well with the number of files being watched. This monitor is the default monitor on systems running inotify-enabled Linux kernels.

5.4.1 Peculiarities

5.4.1.1 Queue Overflow

The inotify monitor may suffer a queue overflow if events are generated faster than they are read from the queue. In any case, the application is guaranteed to receive an overflow notification which can be handled to gracefully recover.

By default, the `fswatch` process is terminated after the notification is sent by throwing an exception. Using the `--allow-overflow` option makes `fswatch` emit a change event of type `Overflow` without exiting.

5.4.1.2 Duplicate Events

The inotify API sends events for the *direct* child elements of a watched directory and it scales pretty well with the number of watched items. For this reason, depending on the number of files to watch, it may sometimes be preferable to non-recursively watch a common parent directory and filter received events rather than adding a huge number of file watches. If recursive watches are used, then duplicate change events will be received:

- One generated by the parent directory of the file that has changed.
- One generated by the file that has changed.

5.5 The Windows monitor

The Windows monitor uses the Windows' `ReadDirectoryChangesW` function for each watched path and asynchronously waits for change events using overlapped I/O. The Windows monitor is the default choice on Windows because it is the best performing monitor on that platform and it is affected by virtually no limitations.

5.5.1 Peculiarities

5.5.1.1 Buffer Overflow

The Windows monitor may suffer a buffer overflow if events are generated faster than they can be stored in the buffer allocated by the operating system when `ReadDirectoryChangesW` is first called on a watched path. Once the buffer has been created, it is never resized and will live until the file handle events are listened upon is closed.

Another source of overflow is the size of the buffer passed to `ReadDirectoryChangesW` by its caller. Unless the one created by Windows, this buffer's size can be tuned by the user. The custom `windows.ReadDirectoryChangesW.buffer.size` property can be used to programmatically set the size of the buffer (in bytes) when `fswatch` is invoked, as shown in the following example where a 4 kilobytes buffer is used:

```
$ fswatch --monitor-property \
```

```
windows.ReadDirectoryChangesW.buffer.size=4096 \
~
```

By default, the `fswatch` process is terminated after the notification is sent by throwing an exception. Using the `--allow-overflow` option makes `fswatch` emit a change event of type `Overflow` without exiting.

5.5.1.2 Directory Watching

The Windows API lets user watch *directory*, not *files*. `fswatch` currently passes path arguments to the underlying monitor as they are: as a consequence, if a path corresponds to a file, the monitor will emit an error and will not be able to watch it.

5.5.1.3 Recursivity

The Windows API will return change events related to a watched directory and any children of its, at any depth. Essentially, the subtree rooted at a directory is *recursively* watched even if the `-r` option is not used explicitly.

5.6 The Poll Monitor

The poll monitor was added as a fallback mechanisms in the cases where no other monitor could be used, including:

- Operating system without any sort of file events API.
- Situations where the limitations of the available monitors cannot be overcome[2].

The poll monitor, available on any platform, only relies on available CPU and memory to perform its task.

5.6.1 Peculiarities

5.6.1.1 Performance Problems

The resource consumption of this monitor increases increases *linearly* with the number of files being watched (the resulting system performance will probably degrade *linearly* or quicker).

The authors' experience indicates that `fswatch` requires approximately 150 MB of RAM memory to observe a hierarchy of 500,000 files with a minimum path length of 32 characters. A common bottleneck of the poll monitor is disk access, since `stat()`-ing a great number of files may take a *huge* amount of time. In this case, the latency (see [Latency], page 19) should be set to a sufficiently large value in order to reduce the performance degradation that may result from frequent disk access; the inotify monitor, in fact,

[2] E.g.: observing a number of files greater than the available file descriptors on a system using the kqueue monitor.

will re-scan *all* the monitored object hierarchy looking for differences *every* time its 'monitoring loop' is repeated.

> **Note:** Using a disk drive with lower latencies may certainly help, although the authors suspect that switching to an operating system with proper file monitoring APIs is a better solution when performance problems with the poll monitors are experienced or when `fswatch` should drive mission-critical processes.

5.6.1.2 Missing Events and Missing Event Flags

Since this monitor periodically checks the state of monitored objects looking for differences, it may miss events happened between one scan and another. Let's suppose, for example, that a file `file` exists at time t_0 when a scan occurs. The poll monitors detects `file` and saves the relevant attributes in memory. `file` is then updated, moved to another place and recreated with the same name. The chain of events[3] occurred to `file` are:

- `Updated`
- `MovedFrom` (or `Deleted`)
- `Created`
- `Link`

At time t_1, another scan runs and the poll monitor detects that the modification date has changed. The poll monitor can only infer that a 'change' has occurred and raises an `Updated` event; other events that would be noticed and raised by other APIs are effectively *lost* since they go unnoticed.

The odds of incurring such a loss is inversely proportional to the latency l: reducing the latency helps alleviating this problem, although on the other hands it also results in linearly increasing resource usage.

5.7 How to Choose a Monitor

`fswatch` already chooses the 'best' monitor for your platform if you do not specify any. However, a specific monitor may be better suited to specific use cases. Please, see Chapter 5 [Monitors], page 25 to get a description of all the available monitors and their limitations.

Usage recommendations are as follows:

- On OS X, use only the FSEvents monitor (which is the default behaviour).
- On Linux, use the inotify monitor (which is the default behaviour).
- If the number of files to observe is sufficiently small, use the kqueue monitor. Beware that on some systems the maximum number of file descriptors that can be opened by a process is set to a very *low* value (values as low as 256 are not uncommon), even if the operating system

[3] The actual chain of events may in fact vary depending on the monitor being used.

may allow a much larger value. In this case, check your OS documentation to raise this limit on either a per process or a system-wide basis.

- If feasible, watch directories instead of watching files. Properly crafting the receiving side of the events to deal with directories may sensibly reduce the monitor resource consumption.
- If none of the above applies, use the poll monitor. The authors' experience indicates that fswatch requires approximately 150 MB of RAM memory to observe a hierarchy of 500,000 files with a minimum path length of 32 characters. A common bottleneck of the poll monitor is disk access, since `stat()`-ing a great number of files may take a huge amount of time. In this case, the latency should be set to a sufficiently large value in order to reduce the performance degradation that may result from frequent disk access.

Appendix A GNU Free Documentation License

Version 1.3, 3 November 2008

0. PREAMBLE

   The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

   This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

   We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

   This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

   A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

   A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

   You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

   You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

   If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled “History”, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled “History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with. . . Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Appendix B Index of Functions

Appendix C Index of Programs

F

R

X

Appendix D Index of Command Line Options

This appendix contains an index of all `fswatch` long command line options. The options are listed without the preceding double-dash.

Index

%

A

B

C

E

F

G

I

K

L

M

O

P

Q

R

S

T

W