With-Editor User Manual

for version 2.7.1

Jonas Bernoulli

Table of Contents

1 Using the With-Editor package

The `With-Editor` package is used internally by Magit when editing commit messages and rebase sequences. It also provides some commands and features which are useful by themselves, even if you don't use Magit.

For information about using this library in you own package, see Chapter 2 [Using With-Editor as a library], page 4.

1.1 Configuring With-Editor

With-Editor tries very hard to locate a suitable `emacsclient` executable, so ideally you should never have to customize the option `with-editor-emacsclient-executable`. When it fails to do so, then the most likely reason is that someone found yet another way to package Emacs (most likely on macOS) without putting the executable on `$PATH`, and we have to add another kludge to find it anyway.

`with-editor-emacsclient-executable` [User Option]

The `emacsclient` executable used as the editor by child process of this Emacs instance. By using this executable, child processes can call home to their parent process.

This option is automatically set at startup by looking in `exec-path`, and other places where the executable could be installed, to find the `emacsclient` executable most suitable for the current Emacs instance.

You should **not** customize this option permanently. If you have to do it, then you should consider that a temporary kludge and inform the Magit maintainer as described in Chapter 3 [Debugging], page 5.

If With-Editor fails to find a suitable `emacsclient` on you system, then this should be fixed for all users at once, by teaching `with-editor-locate-emacsclient` how to do so on your system and system like yours. Doing it this way has the advantage, that you won't have do it again every time you update Emacs, and that other users who have installed Emacs the same way as you have, won't have to go through the same trouble.

Note that there also is a nuclear option; setting this variable to `nil` causes the "sleeping editor" described below to be used even for local child processes. Obviously we don't recommend that you use this except in "emergencies", i.e. before we had a change to add a kludge appropriate for you setup.

`with-editor-locate-emacsclient` [Function]

The function used to set the initial value of the option `with-editor-emacsclient-executable`. There's a lot of voodoo here.

The `emacsclient` cannot be used when using Tramp to run a process on a remote machine. (Theoretically it could, but that would be hard to setup, very fragile, and rather insecure).

With-Editor provides an alternative "editor" which can be used by remote processes in much the same way as local processes use an `emacsclient` executable. This alternative is known as the "sleeping editor" because it is implemented as a shell script which sleeps until it receives a signal.

`with-editor-sleeping-editor` [User Option]

The sleeping editor is a shell script used as the editor of child processes when the `emacsclient` executable cannot be used.

This fallback is used for asynchronous process started inside the macro `with-editor`, when the process runs on a remote machine or for local processes when `with-editor-emacsclient-executable` is `nil`.

Where the latter uses a socket to communicate with Emacs' server, this substitute prints edit requests to its standard output on which a process filter listens for such requests. As such it is not a complete substitute for a proper `emacsclient`, it can only be used as `$EDITOR` of child process of the current Emacs instance.

Some shells do not execute traps immediately when waiting for a child process, but by default we do use such a blocking child process.

If you use such a shell (e.g. `csh` on FreeBSD, but not Debian), then you have to edit this option. You can either replace `sh` with `bash` (and install that), or you can use the older, less performant implementation:

```
"sh -c '\
echo \"WITH-EDITOR: $$ OPEN $0\"; \
trap \"exit 0\" USR1; \
trap \"exit 1\" USR2; \
while true; do sleep 1; done'"
```

This leads to a delay of up to a second. The delay can be shortened by replacing `sleep 1` with `sleep 0.01`, or if your implementation does not support floats, then by using `nanosleep 0.01` instead.

1.2 Using With-Editor commands

This section describes how to use the `with-editor` library *outside* of Magit. You don't need to know any of this just to create commits using Magit.

The commands `with-editor-async-shell-command` and `with-editor-shell-command` are intended as drop in replacements for `async-shell-command` and `shell-command`. They automatically export `$EDITOR` making sure the executed command uses the current Emacs instance as "the editor". With a prefix argument these commands prompt for an alternative environment variable such as `$GIT_EDITOR`.

`with-editor-async-shell-command` [Command]

This command is like `async-shell-command`, but it runs the shell command with the current Emacs instance exported as `$EDITOR`.

`with-editor-shell-command` [Command]

This command is like `async-shell-command`, but it runs the shell command with the current Emacs instance exported as `$EDITOR`. This only has an effect if the shell command is run asynchronously, i.e. when the command ends with `&`.

To always use these variants add this to you init file:

```
(define-key (current-global-map)
  [remap async-shell-command] 'with-editor-async-shell-command)
```

```
(define-key (current-global-map)
  [remap shell-command] 'with-editor-shell-command)
```

Alternatively use the global `shell-command-with-editor-mode`.

`shell-command-with-editor-mode` [Variable]

When this mode is active, then `$EDITOR` is exported whenever ultimately `shell-command` is called to asynchronously run some shell command. This affects most variants of that command, whether they are defined in Emacs or in some third-party package.

The command `with-editor-export-editor` exports `$EDITOR` or another such environment variable in `shell-mode`, `term-mode` and `eshell-mode` buffers. Use this Emacs command before executing a shell command which needs the editor set, or always arrange for the current Emacs instance to be used as editor by adding it to the appropriate mode hooks:

```
(add-hook 'shell-mode-hook  'with-editor-export-editor)
(add-hook 'term-exec-hook   'with-editor-export-editor)
(add-hook 'eshell-mode-hook 'with-editor-export-editor)
```

Some variants of this function exist; these two forms are equivalent:

```
(add-hook 'shell-mode-hook
          (apply-partially 'with-editor-export-editor "GIT_EDITOR"))
(add-hook 'shell-mode-hook 'with-editor-export-git-editor)
```

`with-editor-export-editor` [Command]

When invoked in a `shell-mode`, `term-mode`, or `eshell-mode` buffer, this command teaches shell commands to use the current Emacs instance as the editor, by exporting `$EDITOR`.

`with-editor-export-git-editor` [Command]

This command is like `with-editor-export-editor` but exports `$GIT_EDITOR`.

`with-editor-export-hg-editor` [Command]

This command is like `with-editor-export-editor` but exports `$HG_EDITOR`.

2 Using With-Editor as a library

This section describes how to use the `with-editor` library *outside* of Magit to teach another package how to have its child processes call home, just like Magit does. You don't need to know any of this just to create commits using Magit. You can also ignore this if you use `with-editor` outside of Magit, but only as an end-user.

For information about interactive use and options that affect both interactive and non-interactive use, see Chapter 1 [Using the With-Editor package], page 1.

`with-editor` **&rest** *body* [Macro]

This macro arranges for the `emacsclient` or the sleeping editor to be used as the editor of child processes, effectively teaching them to call home to the current Emacs instance when they require that the user edits a file.

This is essentially done by establishing a local binding for `process-environment` and changing the value of the `$EDITOR` environment variable in that scope. This affects all asynchronous processes started by forms (dynamically) inside BODY.

`with-editor-set-process-filter` *process filter* [Function]

This function is like `set-process-filter` but ensures that adding the new FILTER does not remove the `with-editor-process-filter`. This is done by wrapping the two filter functions using a lambda, which becomes the actual filter. It calls `with-editor-process-filter` first, passing `t` as NO-STANDARD-FILTER. Then it calls FILTER.

3 Debugging

With-Editor tries very hard to locate a suitable `emacsclient` executable, and then sets option `with-editor-emacsclient-executable` accordingly. In very rare cases this fails. When it does fail, then the most likely reason is that someone found yet another way to package Emacs (most likely on macOS) without putting the executable on `$PATH`, and we have to add another kludge to find it anyway.

If you are having problems using `with-editor`, e.g. you cannot commit in Magit, then please open a new issue at `https://github.com/magit/with-editor/issues` and provide information about your Emacs installation. Most importantly how did you install Emacs and what is the output of `M-x with-editor-debug RET`.