Devil Mode

Susam Pal

05 Oct 2023

Devil mode trades your comma key in exchange for a modifier-free editing experience in Emacs. Yes, the comma key! The key you would normally wield for punctuation in nearly every corner of text. Yes, this is twisted! It would not be called the Devil otherwise, would it? If it were any more rational, we might call it something divine, like, uh, the God mode? But alas, there is nothing divine to be found here. Welcome, instead, to the realm of the Devil! You will be granted the occasional use of the comma key for punctuation, but only if you can charm the Devil. But beware, for in this sinister domain, you must relinquish your comma key and embrace an editing experience that whispers wicked secrets into your fingertips!

1 Introduction

Devil mode intercepts our keystrokes and translates them to Emacs key sequences according to a configurable set of translation rules. For example, with the default translation rules, when we type `, x , f`, Devil translates it to `C-x C-f`.

The choice of the comma key (`,`) to mean the control modifier key (`C-`) may seem outrageous. After all, the comma is a very important punctuation both in prose as well as in code. Can we really get away with using `,` to mean the `C-` modifier? It turns out, this terrible idea can be made to work without too much of a hassle. At least it works for me. It might work for you too. If it does not, Devil can be configured to use another key instead of `,` to mean the `C-` modifier. See the section 11.4 and the subsequent sections for a few examples.

A sceptical reader may rightfully ask: If `,` is translated to `C-`, how on earth are we going to insert a literal `,` into the text when we need to? The section 5 answers this. But before we get there, we have some fundamentals to cover. Take the plunge and see what unfolds. Maybe you will like this. Maybe you will not. If you do not like this, you can always retreat to God mode, Evil mode, the vanilla key bindings, or whatever piques your fancy.

2 Notation

A quick note about the notation used in the document: The previous example shows that `, x , f` is translated to `C-x C-f`. What this really means is that the keystrokes `,x,f` is translated to `ctrl+x ctrl+f`. We do not really type any space after the commas. The key `,` is directly followed by the key `x`. However, the key sequence notation used in this document contains spaces between each keystroke. This is consistent with how key sequences are represented in Emacs in general and how Emacs functions like `key-description`, `describe-key`, etc. represent key sequences. When we really need to type a space, it is represented as `SPC`.

3 Install

Devil is available via NonGNU ELPA and MELPA. You may already have a preferred way of installing packages from ELPA/MELPA. If so, install the package named `devil` to get Devil. If you have Emacs 28.1 or a more recent version, it has NonGNU ELPA enabled by default, so you can install Devil quite easily with `M-x package-install RET devil RET` without having to perform any further steps. However, for the sake of completeness, a few very different and basic ways of installing Devil are presented in the next few subsections.

3.1 Install Interactively from MELPA

To install the latest version of Devil from MELPA, perform the following steps:

1. Add the following to the Emacs initialization file (i.e., `~/.emacs` or `~/.emacs.d/init.el` or `~/.config/emacs/init.el`):

   ```
   (require 'package)
   (add-to-list 'package-archives '("melpa" . "https://melpa.org/packages/") t)
   (package-initialize)
   ```

2. Start Emacs with the updated initialization file. Then type these commands:

   ```
   M-x package-refresh-contents RET
   M-x package-install RET devil RET
   ```

3. Confirm that Devil is installed successfully with this command:

   ```
   C-h f devil RET
   ```

4. Enable Devil mode with this command:

   ```
   M-x global-devil-mode RET
   ```

5. Type `, x , f` and watch Devil translate it to `C-x C-f` and invoke the corresponding command.

3.2 Install Automatically from MELPA

Here is yet another basic way to install and enable Devil using Elisp:

```
(require 'package)
(add-to-list 'package-archives '("melpa" . "https://melpa.org/packages/") t)
(package-initialize)
(unless package-archive-contents
  (package-refresh-contents))
(unless (package-installed-p 'devil)
  (package-install 'devil))
(global-devil-mode)
(global-set-key (kbd "C-,") 'global-devil-mode)
```

Now type `, x , f` and watch Devil translate it to `C-x C-f` and invoke the corresponding command. Type `C-,` to disable Devil mode. Type `C-,` again to enable it.

3.3 Install from Git Source

If you prefer obtaining Devil from its Git repository, follow these steps:

1. Clone Devil to your system:

   ```
   git clone https://github.com/susam/devil.git
   ```

2. Add the following to your Emacs initialization:

   ```
   (add-to-list 'load-path "/path/to/devil/")
   (require 'devil)
   (global-devil-mode)
   (global-set-key (kbd "C-,") 'global-devil-mode)
   ```

3. Start the Emacs editor. Devil mode should now be enabled in all buffers. The modeline of each buffer should show the `Devil` lighter.

4. Type `, x , f` and watch Devil translate it to `C-x C-f` and invoke the corresponding command. Type `C-,` to disable Devil mode. Type `C-,` again to enable it.

4 Use Devil

Assuming vanilla Emacs key bindings have not been changed and Devil has not been customised, here are some examples that demonstrate how Devil may be used:

1. Type `, x , f` and watch Devil translate it to `C-x C-f` and invoke the `find-file` command.

2. Type `, p` to move up one line.

3. To move up multiple lines, type `, p p p` and so on. Some Devil key sequences are repeatable keys by default. The repeatable Devil key sequences can be repeated by typing the last key of the Devil key sequence over and over again.

4. Each repeatable key sequence belongs to a repeatable key sequence groups. Like before, type `, p p p` to move the cursor up by a few lines. But then immediately type `n n` to move the cursor down by a couple of lines. Then immediately type `p n b f` to move the cursor up, down, left, and right. The key sequences `, p` and `, n` and `, f` and `, b` form a single repeatable key sequence group. Therefore after we type any one of them, we can repeat that key sequence or any other key sequence in the same group over and over again merely by typing the last character of that key sequence. Typing any other key stops the repetition and the default behaviour of that other key is then observed. Type `C-h v devil-repeatable-keys RET` to see the complete list of all repeatable key sequence groups.

5. Sometimes a repeatable key sequence may be the only key sequence in a repeatable key sequence group. An example of such a key sequence is `, m ^` which translates to `M-^` and joins the current line to the previous line. In a text buffer with multiple lines type `, m ^` to join the current line to the previous line. Then type `^` repeatedly to continue joining lines. Typing any other key stops the repetition.

6. Type `, s` and watch Devil translate it to `C-s` and invoke incremental search.

7. Type `, m x` and watch Devil translate it to `M-x` and invoke the corresponding command. Yes, `, m` is translated to `M-`.

8. Type `, m m s` and watch Devil translate it to `C-M-s` and invoke regular-expression-based incremental search. The key sequence `, m m` is translated to `C-M-`.

9. Type `, u , f` and watch Devil translate it to `C-u C-f` and move the cursor forward by 4 characters.

10. Type `, u u , f` and the cursor moves forward by 16 characters. Devil uses its translation rules and an additional keymap to make this input key sequence behave like `C-u C-u C-f` which moves the cursor forward by 16 characters.

11. Type `, SPC` to type a comma followed by space. This is a special key sequence to make it convenient to type a comma in the text. Note that this sacrifices the use of `, SPC` to mean `C-SPC` which could have been a convenient way to set a mark. See the section 11.3 if you do not want to make this sacrifice.

12. Type `, z SPC` and watch Devil translate it to `C-SPC` and set a mark. Yes, `, z` is translated to `C-` too.

13. Similarly, type `, RET` to type a comma followed by the `enter` key. This is another special key.

14. Type `, ,` to type a single comma. This special key is useful for cases when you really need to type a single literal comma.

15. Type `, h , k` to invoke `devil-describe-key`. This is a special key that invokes the Devil variant of `describe-key` included in vanilla Emacs. When the key input prompt appears, type the Devil key sequence `, x , f` and Devil will display the documentation of the function invoked by this Devil key sequence. Note: The key sequence `, h k` translates to `C-h k` and invokes the vanilla `describe-key`. It is the Devil key sequence `, h , k` that invokes `devil-describe-key`.

5 Typing Commas

Devil makes the questionable choice of using the comma as its activation key. As illustrated in the previous section, typing `, x , f` produces the same effect as typing `C-x C-f`. One might naturally wonder how then we are supposed to type literal commas.

Most often when we edit text, we do not really type a comma in isolation. Often we immediately follow the comma with a space or a newline. This assumption usually holds good while editing regular text. However, this assumption may not hold in some situations, like while working with code when we need to add a single comma at the end of an existing line.

In scenarios where the above assumption holds good, typing `, SPC` inserts a comma and a space. Similarly, typing `, RET` inserts a comma and a newline.

In scenarios where we do need to type a single comma, type `, ,` instead.

Note that you could also type a single comma with `, q ,` which translates to `C-q ,` and inserts a literal comma. The Emacs key sequence `C-q` invokes the command `quoted-insert` which inserts the next input character. The `, ,` special key sequence is probably easier to type than this.

Also, it is worth mentioning here that if all this fiddling with the comma key feels clumsy, we could always customise the Devil key to something else

that feels better. We could also disable Devil mode temporarily and enable it again later with `C-,` as explained in section 3.

6 Devil Reader

The following points briefly describe how Devil reads Devil key sequences, translates them to Emacs key sequences, and runs commands bound to the key sequences:

1. As soon as the Devil key is typed (which is `,` by default), Devil wakes up and starts reading Devil key sequences. Type `C-h v devil-key RET` to see the current Devil key.

2. After each keystroke is read, Devil checks if the key sequence accumulated is a special key. If it is, then the special command bound to the special key is executed immediately. Note that this step is performed before any translation rules are applied to the input key sequence. This is how the Devil special key sequence `, SPC` inserts a comma and a space. Type `C-h v devil-special-keys RET` to see the list of special keys and the commands bound to them.

3. If the key sequence accumulated so far is not a special key, then Devil translates the Devil key sequence to a regular Emacs key sequence. If the regular Emacs key sequence turns out to be a complete key sequence and some command is found to be bound to it, then that command is executed immediately. This is how the Devil key sequence `, x , f` is translated to `C-x C-f` and the corresponding binding is executed. If the translated key sequence is a complete key sequence but no command is bound to it, then Devil displays a message that the key sequence is undefined. Type `C-h v devil-translations RET` to see the list of translation rules.

4. After successfully translating a Devil key sequence to an Emacs key sequence and executing the command bound to it, Devil checks if the key sequence is a repeatable key sequence. If it is found to be a repeatable key sequence, then Devil sets a transient map so that the repeatable key sequences that belong to the same group as the typed Devil key sequence can be invoked merely by typing the last character of the input key sequence. This is how `, p p p f f` moves the cursor up by three lines and then by two characters forward. Type `C-h v devil-repeatable-keys RET` to see the list of repeatable Devil key sequences.

The variables `devil-special-keys`, `devil-translations`, and `devil-repeatable-keys` may contain keys or values with the string `%k` in them. This is a placeholder for `devil-key`. While applying the special keys, translation rules, or repeat

rules, each %k is replaced with the actual value of devil-key before applying the rules.

7 Translation Mechanism

The following points provide an account of the translation mechanism that Devil uses in order to convert a Devil key sequence entered by the user to an Emacs key sequence:

1. The input key vector read from the user is converted to a key description (like the string produced by functions like describe-key and key-description). For example, if the user types ,x,f it is converted to , x , f.
2. Now the resulting key description is translated with simple string replacements. If any part of the string matches a key in devil-translations, then it is replaced with the corresponding value. For example, , x , f is translated to C- x C- f. Then Devil normalises the result to C-x C-f by removing the stray spaces after the modifier keys.
3. If the simple string based replacement discussed in the previous point leads to an invalid Emacs key sequence, it skips the replacement that causes the resulting Emacs key sequence to become invalid. For example , m m , results in C-M-C- after the simple string replacement because the default translation rules replace the leading , m m with C-M- and the trailing , with C-. However, C-M-C- is an invalid key sequence, so the replacement of the trailing , to C- is skipped. Therefore, the input , m m , is translated to C-M-, instead.
4. Finally, Devil looks for key chords in the key sequence that contain both the C- modifier and an uppercase letter. If such a key chord occurs, then it replaces the uppercase letter with its shifted form, e.g., , m m V first translates to C-M-V according to the previous points and then the result is translated to C-M-S-v according to this point.

8 Default Translation Rules

By default, Devil supports a small but peculiar set of translation rules that can be used to avoid modifier keys while typing various types of key sequences. See C-h v devil-translations RET for the translation rules. Here are some examples that demonstrate the default translation rules. The obvious ones are shown first. The more peculiar translations come later in the table. The concluding paragraph of this subsection offers a guide on how to gradually and gently adopt these key sequences into your daily routine.

Input	Translated	Remarks
`, s`	`C-s`	Rule 1: `,` is replaced with `C-`
`, m x`	`M-x`	Rule 2: `, m` is replaced with `M-`
`, [ x`	`C-[ x`	equivalent to `M-x`
`, m m s`	`C-M-s`	Rule 3: `, m m` is replaced with `C-M-`
`, m ,`	`M-,`	Rule 4: `, m ,` is replaced with `M-,`
`, m z m`	`M-m`	Rule 5: `, m z` is replaced with `M-` too
`, c , ,`	`C-c ,`	Rule 6: `, ,` is replaced with `,`
`, z SPC`	`C-SPC`	Rule 7: `, z` is replaced with `C-` too
`, z z`	`C-z`	ditto
`, z ,`	`C-,`	ditto

Note how we cannot use `, SPC` to set a mark because that key sequence is already reserved as a special key sequence in `devil-special-keys`. In order to conveniently set a mark, Devil translates `, z` to `C-` too, so that we can type `, z SPC` and have Devil translate it to `C-SPC`.

Also, note that while `, m` may be used to type `M-` we have `, [` as yet another way to type a key sequence that contains `M-` because `, [` translates to `C-[` and `C-[ <key>` is equivalent to `ESC <key>` which in turn is equivalent to `M-<key>`.

The default translation examples presented in the table above look weirder and weirder as we go down the table. But they are not as arbitrary as they might initially appear to be. They are arranged in such a way that overall, we get the following effect:

- Devil translates the input `,` to `C-`. Similarly it translates `, m` to `M-` and `, m m` to `C-M-`.
- When we really want to type the Devil key `,` we need to double type it in the Devil key sequence. Doubling the special character serves as an escape mechanism to avoid the special meaning of the Devil key and get its literal form instead.
- Now since `, ,` translates to `,` we need another escape mechanism to type `C-,`. Typing `z` in between serves as this escape mechanism, i.e., within a Devil key sequence `, z ,` translates to `C-,`.
- Similarly since `, m m` translates to `C-M-` we need an escape mechanism to type `M-m`. Again, typing `z` in between serves as this escape mechanism, i.e., `, m z m` translates to `M-m`.

Here is a gentle guide to adopting these key sequences: For beginners using Devil, it is not necessary to memorise all of them right away. Understanding that `,` translates to `C-` and `, m` translates to `M-` is sufficient to begin. Subsequently, learning that `, m m` translates to `C-M-` unlocks several more key sequences like `, m m s` (`C-M-s`), `, m m f` (`C-M-f`), etc. As you encounter

more key sequences that are not covered by these initial rules, revisit the above table to pick up new translation rules and adopt them in your day-to-day usage of Devil.

9 Describe Devil Key

Devil offers a command named `devil-describe-key` that can be used to describe a Devil key sequence. It works similarly to the `describe-key` command of vanilla Emacs that can be invoked with `C-h k`. The `devil-describe-key` command can be invoked with the special key sequence `, h , k`. Type `, h , k` and a prompt appears to read a key sequence. Type any Devil key sequence, say, `, x , f` and Devil immediately shows the documentation for the function invoked by this key sequence.

Note that `, x , f` (`devil-describe-key`) can also be used to look up documentation for vanilla Emacs key sequences like `C-x C-f`.

Also note that the Devil key sequence is `, h k` is still free to invoke `C-h k` (`describe-key` of vanilla Emacs).

10 Bonus Key Bindings

Devil adds the following additional key bindings only when Devil is enabled globally with `global-devil-mode`:

- Adds the Devil key to `isearch-mode-map`, so that Devil key sequences work in incremental search too.
- Adds `u` to `universal-argument-more` to allow repeating the universal argument command `C-u` simply by repeating `u`.

As mentioned before these features are available only when Devil is enabled globally with `global-devil-mode`. If Devil is enabled locally with `devil-mode`, then these features are not available.

11 Custom Configuration Examples

In the examples presented below, the `(require 'devil)` calls may be omitted if Devil has been installed from a package archive like ELPA or MELPA. There are appropriate autoloads in place in the Devil package that would ensure that it is loaded automatically on enabling Devil mode. However, the `require` calls have been included in the examples below for the sake of completeness.

11.1 Local Mode

While the section 3 shows how we enable Devil mode globally, this section shows how we can enable it locally. Here is an example initialization code that enables Devil locally only in text buffers.

```
(require 'devil)
(add-hook 'text-mode-hook 'devil-mode)
(global-set-key (kbd "C-,") 'devil-mode)
```

This is not recommended though because this does not provide a seamless Devil experience. For example, with Devil enabled locally in a text buffer like this, although we can type `, x , f` to launch the `find-file` minibuffer, we cannot use Devil key sequences in the minibuffer. Further the special keymaps described in the previous section work only when Devil is enabled globally.

11.2 Custom Appearance

The following initialization code shows how we can customise Devil to show a Devil smiley in the modeline and in the Devil prompt.

```
(require 'devil)
(setq devil-lighter " \U0001F608")
(setq devil-prompt "\U0001F608 %t")
(global-devil-mode)
(global-set-key (kbd "C-,") 'global-devil-mode)
```

11.3 Reclaim , SPC to Set Mark

The default configuration for special keys reserves `, SPC` to insert a literal comma followed by space. This default makes it easy to type comma in various contexts. However, this means that `, SPC` does not translate to `C-SPC`. Therefore `, SPC` cannot be used to set mark. Instead, the default translation rules offer `, z SPC` as a way to set mark.

If you would rather set mark using `, SPC` and you are happy with typing the special key `, ,` to insert a literal comma, then use the following configuration:

```
(require 'devil)
(global-devil-mode)
(global-set-key (kbd "C-,") 'global-devil-mode)
(assoc-delete-all "%k SPC" devil-special-keys)
```

This removes the special key `, SPC` from `devil-special-keys` so that it is now free to be translated to `C-SPC` and invoke `set-mark-command`.

11.4 Custom Devil Key

The following initialization code shows how we can customise Devil to use a different Devil key.

```
(require 'devil)
(global-devil-mode)
(global-set-key (kbd "C-;") 'global-devil-mode)
(devil-set-key (kbd ";"))
```

The above example sets the Devil key to the semicolon, perhaps another dubious choice for the Devil key. With this configuration, we can use `; x ; f` and have Devil translate it to `C-x C-f`.

11.5 Yet Another Custom Devil Key

The following initialization code shows how we can customise Devil to use yet another different Devil key.

```
(require 'devil)
(global-devil-mode)
(global-set-key (kbd "C-<left>") 'global-devil-mode)
(devil-set-key (kbd "<left>"))
(dolist (key '("%k SPC" "%k RET" "%k <return>"))
  (assoc-delete-all key devil-special-keys))
```

The above example sets the Devil key to the left arrow key. With this configuration, we can use `<left> x <left> f` and have Devil translate it to `C-x C-f`. We can type the special key `<left> <left>` to produce the same effect as the original `<left>`.

The above example removes some special keys that are no longer useful. In particular, `<left> SPC` is no longer reserved as a special key, so we can use it now to set a mark.

11.6 Multiple Devil Keys

While this package provides the comma (`,`) as the default and the only Devil key, nothing stops you from extending the mode map to support multiple Devil keys. Say, you decide that in addition to activating Devil with `,` which also plays the role of `C-`, you also want to activate Devil with `.` which must now play the role of `M-`. To achieve such a result, you could use this initialization code as a starting point and then customise it further based on your requirements:

```
(require 'devil)
(global-devil-mode)
(define-key devil-mode-map (kbd ".") #'devil)
```

```
(add-to-list 'devil-special-keys `(". ." . ,(devil-key-executor ".")))
(setq devil-translations '((", z" . "C-")
                           (". z" . "M-")
                           (", ," . ",")
                           (". ." . ".")
                           ("," . "C-")
                           ("." . "M-")))
```

With this configuration, we can type `, x , f` for `C-x C-f` like before. But now we can also type `. x` for `M-x`. Similarly, we can type `, . s` for `C-M-s` and so on. Also `, ,` inserts a literal comma and `. .` inserts a literal dot. Further we can type `, z ,` to get `C-,` and `. z .` to get `M-.`.

Note that by default Devil configures only one activation key (`,`) because the more activation keys we add, the more intrusive Devil becomes during regular editing tasks. Every key that we reserve for activating Devil loses its default function and then we need workarounds to somehow invoke the default function associated with that key (like repeating `.` twice to insert a single `.` in the above example). Therefore, it is a good idea to keep the number of Devil keys as small as possible.

11.7 Make All Keys Repeatable

By default Devil has a small list of key sequences that are considered repeatable. This list is defined in the variable `devil-repeatable-keys`. Type `C-h v devil-repeatable-keys RET` to view this list. For example, consider the repeatable key sequence group `("%k p" "%k n" "%k f" "%k b")` in this list. Assuming that the default Devil and Emacs key bindings have not been changed, this means that after we type `, p` and move the cursor to the previous line, we can repeat this operation by typing `p` over and again. We can also immediately type `f` to move the cursor right by one character. The repetition occurs as long as the last character of any repeatable key sequence in the group is typed again. Typing any other key stops the repetition and the default behaviour of the other key is then observed.

It is possible to make all key sequences repeatable by setting the variable `devil-all-keys-repeatable` to `t`. Here is an example configuration:

```
(require 'devil)
(setq devil-all-keys-repeatable t)
(global-devil-mode)
```

With this configuration, the repeatable key sequence groups still function as described above. However, in addition to that now all other Devil key sequences that end up executing Emacs commands also become repeatable, i.e., any Devil key sequence that does not belong to `devil-all-keys-repeatable`

but invokes an Emacs command is now repeatable and it can be repeated by merely repeating the last character of the key sequence.

Note that only Devil key sequences that get translated to a regular Emacs key sequence and result in the execution of an Emacs command can be repeatable. The special keys defined in `devil-special-keys` are never repeatable.

11.8 Interaction with Repeat Mode

Repeatable keys in Devil function somewhat like `repeat-mode` introduced in Emacs 28.1. Here is an example configuration that disables repeatable keys in Devil and shows how to use `repeat-mode` instead to define repeatable commands.

```
(require 'devil)
(global-devil-mode)
(setq devil-repeatable-keys nil)

(defvar movement-repeat-map
  (let ((map (make-sparse-keymap)))
    (define-key map (kbd "p") #'previous-line)
    (define-key map (kbd "n") #'next-line)
    (define-key map (kbd "b") #'backward-char)
    (define-key map (kbd "f") #'forward-char)
    map))

(dolist (cmd '(previous-line next-line backward-char forward-char))
  (put cmd 'repeat-map 'movement-repeat-map))

(repeat-mode)
```

Now if we type `C-p` to move the cursor up by one line, we can repeat it by merely typing `p` again and we can also type or repeat `n`, `b`, or `f`, to move the cursor down, left, or right respectively.

Repeat mode works fine with Devil too, so with the above configuration, when we type `, p` to move the cursor to the previous line, we can type or repeat `p`, `n`, `b`, or `f` to move the cursor up, down, left, or right again.

We do not really need to disable Devil's repeatable keys while using repeat mode. Both can be enabled together. However, the results can be surprising due to certain differences between the two. For example, consider the following configuration:

```
(require 'devil)
(global-devil-mode)
```

```
(defvar movement-repeat-map
  (let ((map (make-sparse-keymap)))
    (define-key map (kbd "p") #'previous-line)
    (define-key map (kbd "n") #'next-line)
    map))

(dolist (cmd '(previous-line next-line))
  (put cmd 'repeat-map 'movement-repeat-map))

(repeat-mode)
```

Now both Devil repeatable keys and repeat mode are active. If we now type `, p` we can repeat `p` and `n` to move the cursor up and down. Repeat mode makes this repetition possible. Additionally, after typing `, p` we can also type or repeat `b` and `f` to move the cursor left and right. Devil makes this repetition possible. We can tell the difference between repeat mode handling repeatable commands and Devil mode handling repeatable keys by looking at the echo area. When we repeat `p` which is handled by repeat mode, we see a message "Repeat with p, n" in the echo area. But when we repeat `b` which is handled by Devil, we see no such message; Devil sets up repeatable keys silently.

11.9 Comparison with Repeat Mode

The previous section demonstrates how much of what Devil accomplishes with its support for repeatable key sequences can also be accomplished with `repeat-mode` that comes out of the box in Emacs 28.1 and later versions.

However, there is a crucial difference between Devil's repeatable keys and `repeat-mode`. Repeat mode provides repeatable *commands* but Devil supports repeatable *keys*. This different is crucial and arguably makes repeatable key sequences easier to configure in Devil. To demonstrate the difference, let us consider the key sequence `M-e`. The command `forward-sentence` is bound to it by default in the global map. However, in Org mode, the command `org-forward-sentence` is bound to it. The corresponding Devil key sequence is `, m e` and this is a repeatable key sequence in Devil. Therefore, we can type `, m e` followed by `e e e` and so on to move the cursor forward by multiple sentences in text mode as well as in Org mode.

To emulate the same behaviour using repeat mode, we need a configuration like this:

```
(require 'devil)
(global-devil-mode)
(setq devil-repeatable-keys nil)
```

```
(defvar forward-sentence-repeat-map
  (let ((map (make-sparse-keymap)))
    (define-key map (kbd "e") #'forward-sentence)
    map))

(defvar org-forward-sentence-repeat-map
  (let ((map (make-sparse-keymap)))
    (define-key map (kbd "e") #'org-forward-sentence)
    map))

(put #'forward-sentence 'repeat-map 'forward-sentence-repeat-map)
(put #'org-forward-sentence 'repeat-map 'org-forward-sentence-repeat-map)

(repeat-mode)
```

Note how we need to configure repeat mode for both commands that are bound to `M-e`. With the above configuration, we can now type `, m e` followed by `e e e` to move forward by multiple sentences in both text mode as well as Org mode. However, we can never be sure if we missed configuring repeat mode for some other command that might be bound to `M-e` in some mode. For example, in C mode, the command `c-end-of-statement` is bound to `M-e`. The above configuration is no good for repeating this command by typing `e e e`.

Devil, however, can repeat the command bound to `M-e` in any mode. Devil does not merely make the command bound to it in a particular mode repeatable. Instead Devil makes the key sequence `, m e` itself repeatable. Therefore, with Devil's own support for repeatable key sequences, we can type `, m e` and then `e e e` to repeat the command bound to `M-e` regardless of which mode is active or which command is bound to this key sequence.

12 Why?

Why go to the trouble of creating and using something like this? Why not just remap `caps lock` to `ctrl` like every other sane person does? Or if it is so important to avoid modifier keys, why not use something like God mode or Evil mode?

Well, for one, both God mode and Evil mode are modal editing modes. Devil, on the other hand, retains the non-modal editing experience of Emacs.

Devil mode began as a fun little experiment. From the outset, it was clear that using something as crucial as the comma for specifying the modifier key is asking for trouble. However, I still wanted to see how far I could go

with it. It turned out that in a matter of days, I was using it full-time for all of my Emacs usage.

This experiment was partly motivated by Macbook keyboards which do not have a `ctrl` key on the right side of the keyboard. Being a touch-typist myself, I found it inconvenient to type key combinations like `C-x`, `C-s`, `C-r`, `C-d`, `C-f`, `C-w`, `C-a`, `C-e`, etc. where both the modifier key and the modified key need to be pressed with the left hand fingers. I am not particularly fond of remapping `caps lock` to behave like `ctrl` because that still suffers from the problem that key combinations like `C-x`, `C-a` require pressing both the modifier key and the modified key with the left hand fingers. I know many people remap both their `caps lock` and `enter` to behave like `ctrl`. While I think that is a fine solution, I was not willing to put up with the work required to make that work seamlessly across all the various operating systems I work on.

What began as a tiny whimsical experiment a few years ago turned out to be quite effective, at least to me. I like that this solution is implemented purely as Elisp and therefore does not have any external dependency. I am sharing this solution in the form of a minor mode, just in case, there is someone out there who might find this useful too.

13 Comparison with God Mode

God mode provides a modal editing experience but Devil does not. Devil has the same underlying philosophy as that of God mode, i.e., the user should not have to learn new key bindings. However, Devil does not have a hard separation between insert mode and command mode like God mode has. Instead, Devil waits for an activation key (`,` by default) and as soon as it is activated, it intercepts and translates keys, runs the corresponding command, and then gets out of the way. So Devil tries to retain the non-modal editing experience of vanilla Emacs.

Now it is worth mentioning that some of this non-modal editing experience can be reproduced in god-mode too using its `god-execute-with-current-bindings` function. Here is an example:

```
(global-set-key (kbd ",") #'god-execute-with-current-bindings)
```

With this configuration, God mode translates `, x f` to `C-x C-f`. Similarly `, g x` invokes `M-x` and `, G s` invokes `C-M-x`. This provides a non-modal editing experience in God mode too. However, this experience does not extend seamlessly to minibuffers. Devil does extend its Devil key translation to minibuffers.

Further note that in God mode the `ctrl` modifier has sticky behaviour,

i.e., the modifier remains active automatically for the entire key sequence. Therefore in the above example, we type `,` only once while typing `, x f` to invoke `C-x C-f`. However, this sticky behaviour implies that we need some way to disambiguate between key sequences like `C-x C-f` (`find-file`) and `C-x f` (`set-fill-column`). God mode solves this by introducing `SPC` to deactivate the modifier, e.g., `, x f` translates to `C-x C-f` but `, x SPC f` translates to `C-x f`. Devil does not treat the modifier key as sticky which leads to simpler key sequences at the cost of a little additional typing, i.e., `, x , f` translates to `C-x C-f` and `, x f` translates to `C-x f`.

To summarize, there are primarily four things that Devil does differently:

- Provide a non-modal editing experience from the outset.
- Seamlessly extend the same editing experience to minibuffer, incremental search, etc.
- Translate key sequences using string replacements. This allows for arbitrary and sophisticated key translations for the adventurous.
- Choose non-sticky behaviour for the modifier keys.

These differences could make Devil easier to use than God mode for some people but clumsy for other people. It depends on one's tastes and preferences.

14 Frequently Asked Questions

1. Why was the comma (`,`) chosen as the default Devil key? Isn't the semicolon (`;`) a better choice since it belongs to the home row?

   Opinions vary. As the author and maintainer of this minor mode, I made a choice to use the comma as the default Devil key. Although, the semicolon belongs to the home row on most keyboards and the comma does not, I find the vertical movement to reach the comma key with the long finger more convenient than the horizontal movement necessary to reach the semicolon with the little finger.

   As a touch typist, my fingers rest on the eight home row keys when idle. The horizontal movement necessary to type the semicolon leads to a significant angular movement of the wrist. Curling my long finger to reach the comma key helps me avoid this wrist strain. If you do not like this default, it is quite easy to customise the Devil key to be the semicolon or any other key of your choice. See the section 11.4 and the subsequent sections to learn how to do this.

2. I am happy with typing `, ,` every time, I need to type a comma. Can I free up `, SPC` to invoke `set-mark-command`?

Yes, this can be done by removing the special key `, SPC` from `devil-special-keys`. See the section 11.3 to find out how to do this.

3. Can I make the Devil key sticky, i.e., can I type `, x f` instead of `, x , f` to invoke `C-x C-f`?

   Devil does not support sticky keys. Say, Devil were to translate `, x f` to `C-x C-f`, how would we invoke `C-x f` then? We need some way to disambiguate between `C-x C-f` and `C-x f`. Different tools take different approaches to disambiguate the two key sequences. God-mode translates `x f` to `C-x C-f` and `x SPC f` to `C-x f`, i.e., God-mode treats the `C-` modifier as sticky by default but when we want to make it non-sticky, we need to type `SPC` in god-mode. This makes some key sequences like `C-x C-f` shorter to type but some other key sequences like `C-x f` longer to type.

   Devil treats the Devil key as non-sticky, so that there is no need for additional peculiar rules to switch between sticky and non-sticky behaviour to disambiguate key sequences like `C-x C-f` and `C-x f`. With Devil `, x , f` translates to `C-x C-f` and similarly `, x f` translates to `C-x f`. The translation rules are simpler at the cost of a little additional typing in some cases. In most such cases, Devil requires typing an additional comma that one could have avoided if the comma were sticky. However, in other cases, Devil eliminates the need to type an extra key to make the modifier key non-sticky.

4. Are there some things that are easier to do with Devil than god-mode?

   Devil is not necessarily easier than god-mode. It is different. Preferences vary, so some may find Devil easier to use while some others may find god-mode easier to use. See the section 13 for more details on the differences between the two modes.

15 Conclusion

Devil is a minor mode to translate key sequences. Devil utilises this translation capability to provide a modifier-free editing experience and it does so without resorting to modal-editing. Devil retains the non-modal editing of vanilla Emacs. This mode was written as a quirky experiment to make it easier to use Emacs without modifier keys. However, the resulting mode turned out to be quite convenient to use, in general. You might find Devil comfortable. Or you might find Devil to be a terrible idea. It is also possible that you might find Devil useful but intrusive. In such cases, there are plenty of customisable options that you can modify to configure Devil according to your preferences. If you need any help or if you find any issues, please create an issue at https://github.com/susam/devil/issues.