mindstream

Version 8.17

July 31, 2025

If you've ever created throwaway files named `1.txt` or `blah.py` or `test3.rkt` to quickly write down some thoughts, try out a new idea, or prototype a new software tool, then you've already experienced the inconvenience of having to first come up with a name for these throwaway files, a small obstruction that is sometimes enough to snuff out that momentary creative spark that might have become a worthy conflagration. You've also probably experienced the cost of not creating such files, when just as the quick and anonymous freewriting session started to grow into something you'd want to keep around, an application error or computer crash caused you to lose it all and be more discouraged than when you began.

Regardless of what you're trying to do, it begins with writing. Writing text, writing code, writing thoughts. Writing isn't just a way to express yourself, but a way to think.

Mindstream removes the barriers so that you can start writing immediately, and so that you can feel secure enough to play with the things you write, knowing that you won't ever lose anything you care about, and at the same time, never have to think again about the things that have served their purpose in the moment.

Mindstream is a lightweight tool that leaves most of the heavy lifting to other packages (including major modes) and technologies (such as Git). It simply uses these together to augment your existing workflows to fill an unmet need.

Every mindstream session begins from a template (an ordinary folder) that you provide, and evolves through the stages of your creative process. The session itself is stored as an ordinary folder in a unique Git repository at a temporary location on disk. This repository is versioned by commits representing your writing process bounded at natural points – by default, the points at which your buffer is saved (whether explicitly by you or implicitly on running a command like `racket-run`). In this way, Mindstream saves you the trouble of coming up with extraneous names (e.g. `draft1.tex`, `draft2.tex`, ..., `draft_final2.tex`, ...) and allows you to focus on the task at hand. You can save and load these sessions, too, and pick up right where you left off, allowing quick freewriting sessions to organically grow into robust creative works.

Typical uses of this package are for early stages of prototyping in a software project, or for exploratory programming to understand a new idea, tool, or technology. It's also great for just taking quick notes or freewriting blog posts or content in authoring settings in general.

Contents

1 Installation

Mindstream is on MELPA, so you can install it in the usual way (either via Straight.el (recommended) or Emacs's built-in package.el), assuming you have MELPA in your configured list of package archives.

Place the following config somewhere in your `.emacs.d`:

```
(use-package mindstream
  :config
  (mindstream-mode))
```

`(mindstream-mode)` initializes the package, providing global keybindings that allow you to enter Mindstream sessions from anywhere, and registering hooks that allow sessions to be versioned when buffers are saved.

2 Usage

1. Run `mindstream-new` (default: `C-c , n`) to start a session.
2. Write!

Save the file at your regular cadence after making a few changes to it. If you pull up a Magit window, you'll notice that there is a distinct commit recorded in the underlying Git repo for your session each time that you saved the file. You can also create new files at the same path, and changes to any of them would also be similarly tracked.

You may notice that there is only one template available to use for your first session. This is because Mindstream creates a simple text template in `~/.emacs.d/mindstream/templates/` if it doesn't find any there. You may want to add more templates that are relevant for you. Let's see how to do that.

2.1 Adding New Session Templates

Mindstream doesn't include any templates out of the box, so you'll probably want to create some for standard sessions you are likely to need, for instance, for programming in your favorite language (perhaps Racket?), or just freewriting some text for your next great novel, following in the keystrokes of Emacs octopuses like Neal Stephenson.

To add a new template, visit `mindstream-template-path` (default: `"~/.emacs.d/mindstream/templates/"`) in your file manager of choice (e.g. Emacs's `dired`, or just a command line), and create a folder there with the name of your template (e.g. `racket`). The folder should contain an ordinary file (or multiple files) with the appropriate extension (e.g. `.rkt` – it could be anything at all that you typically use Emacs to edit). This template will now be available as an option in `mindstream-new`.

2.2 Saving Sessions

You can also save scratch sessions that you'd like to keep by using `mindstream-save-session` (default binding: `C-c , C-s`). This simply clones the session's Git repo to a more permanent and familiar path that you indicate (as opposed to the anonymous session path which is assumed to be ephemeral and defaults to `~/mindstream/anon`), thus preserving the entire session history, allowing it to be navigated and even resumed at any time in the future.

2.3 Entering Sessions Even More Quickly

`mindstream-enter-anonymous-session` (default: `C-c , b`) will take you immediately to an anonymous session for the current major mode, without asking you any questions. If an anonymous session already exists, it will take you there rather than create a new one. In creating a new session, it will use the first template it finds that is recognizable to your current major mode.

If you've got more than one template for a particular major mode, you may want to indicate which one is preferred rather than leave it to chance or accidents of alphabetical order. You can do this by associating each major mode with the name of the preferred template. For example:

```
:custom
...
(mindstream-preferred-template '(racket-mode "racket"))
```

This customization is only relevant when using `mindstream-enter-anonymous-session`, as you would select the template yourself when using `mindstream-new`.

See §4 "Design" to learn more about anonymous sessions.

2.4 Archiving Sessions

Anonymous sessions at `mindstream-path` are considered *active*, that is, they are sessions you are currently in the middle of. Anonymous sessions that have served their purpose, and which you do not save as named sessions, are *archived* instead, which simply moves them to `mindstream-archive-path`. These sessions will retain their filing under folders named for their creation date, under folders named for the template from which the session was begun.

```
anon-or-archive-path/
  python/
    2024-09-02/
      b6c86878fd8bcddfef3c7b4781f8a3b6694e72b7
      ...
  text/
    2024-09-08/
      2b31e1f7e58ab273709bad1bf47888fb523c49af
      b914a595b8d2ad567d4e14ab128570207742ce06
      ...
  ...
```

This path will accumulate hundreds, or thousands, of sessions over time. They are always there and conveniently filed if you need to refer to them, and can serve as a detailed log of your thoughts and work over time. See §5.5 "Choosing an Archive Path" for some further considerations regarding the archive.

Sessions are archived either automatically, depending on your customization of session persistence and uniqueness, or manually by you on demand.

You can manually archive a session using `mindstream-archive` (default binding: `C-c , a`).

2.5 Live Mode!

Live mode configures Mindstream to automatically take some action that you indicate whenever there is a pause (by default, 1.5 seconds) in typing. Typically, this is used in

programming settings to trigger evaluation of the buffer in an accompanying runtime environment.

Live mode is configured by associating each major mode with a desired action to take for sessions in that mode.

For example, use the following config to evaluate your buffer "live" while in Racket Mode:

```
:custom
...
(mindstream-live-action '(racket-mode racket-run))
```

You can "go live" in any Mindstream session with `M-x mindstream-go-live` (default: `C-c , C-l`). If no live action is configured for the major mode, it will simply use the default action of saving the buffer.

Currently, live mode is only supported for individual buffers rather than for the session as a whole, so you would need to "go live" in each session buffer individually.

Go offline with `M-x mindstream-go-offline` (default: `C-c , C-o`).

2.6 Mindstream Anywhere

If you have an existing, ordinary file or directory that you were working on at some point, and if you want to continue working on it in a mindstream session, that's easy enough to do. Simply follow these steps:

1. Create a new folder (give it a representative name, as you would any Mindstream session) and move the file(s) into it.
2. At the command line in that folder, run `git init`.
3. Open the file in Emacs in the usual way and `M-x mindstream-begin-session`.

Mindstream sessions are just ordinary Git repositories. If you wanted to, you could use Mindstream in any Git repo simply by `M-x mindstream-begin-session` after opening a file in the repo, but this isn't a well-supported use case for the moment (e.g. it would result in a lot of commits, and you would most likely want to manually squash them). For now, if you are interested in mindstreaming an existing repo, try out this workflow by Noboru Ota.

2.7 Explore

Try `M-x mindstream- ...` to see all the available interactive commands. These are also included as keybindings in a minor mode, `mindstream-mode`, which allows you to enter a Mindstream session from anywhere, and contains useful commands for active sessions like saving the session, "going live," and so on.

Mindstream commands are bound by default under the prefix `C-c , ...`. You can view all Mindstream commands by using Emacs's `C-h` introspection with this prefix, as in `C-c , C-h`.

3 Customization

As each Mindstream session uses a specific major mode, it inherits all of the customizations you already have (and any that you decide to add) for that mode. There is typically nothing special you need to do beyond this for Mindstream to work seamlessly with all of your workflows when using these modes.

For instance, one common use of Mindstream is as a scratch buffer with Racket Mode. Racket Mode users sometimes like to have a dedicated REPL to view the output of code they write in a particular buffer, instead of reusing a REPL shared across all buffers. If you're a Racket Mode user, whatever customization you've chosen here would apply to Mindstream session buffers just as they would any buffer, and your Racket Mode sessions may or may not have a dedicated REPL depending on how you've customized this for Racket Mode generally.

But if you happen to want to use a different customization for Mindstream session buffers in a certain major mode than you prefer generally for that major mode, advising the `mindstream-new` function could be one way to achieve that. For instance, for the customization we have been talking about:

```
(advice-add 'mindstream-new
            :after
            (lambda (&rest _args)
              (setq-local racket-repl-buffer-name "*scratch -
 Racket REPL*")))
```

3.1 Persistent Sessions

If you would like anonymous sessions to persist across Emacs restarts, set `mindstream-persist` to `t`. By default, they are archived on startup instead. For example, you could put this in the `:custom` section of your `use-package` declaration:

```
:custom
...
(mindstream-persist t)
```

3.2 Multiple Concurrent Anonymous Sessions

By default, starting a new anonymous session for a template (via `mindstream-new`) *archives* any existing anonymous session for that template (leaving sessions for other templates alone). If you would like to support more than one active anonymous session at a time per template instead, set `mindstream-unique` to `nil`. For example, you could put this in the `:custom` section of your `use-package` declaration:

```
:custom
...
(mindstream-unique nil)
```

4 Design

Mindstream structures your workflow in sessions, which are version-controlled files. When you first start a session it begins as anonymous, meaning that it doesn't have a name. If the session develops into something worth keeping, you can save it to a preconfigured (or any) location on disk by giving the session a name. A session is stored as a version-controlled folder. With that in mind, here are some properties of the design:

1. Depending on `mindstream-unique`, there may be only one anonymous session active at any time per template, or there may be more than one.
2. Saving an anonymous session turns it into a named session. Named sessions work the same as anonymous sessions aside from having a name and being in a permanent location on disk. A new anonymous session could be started at any time via `mindstream-new`.
3. New sessions always begin anonymous.
4. Anonymous sessions may persist across Emacs restarts, depending on `mindstream-persist`.
5. Named sessions may be loaded without interfering with any active anonymous sessions.
6. Any number of named sessions could be active at the same time. Sessions are self-contained and independent.

5 Tips

5.1 Magit

Mindstream sessions are stored as Git repos, so you can use standard Git tools as you might with any repo, including Magit.

Magit is useful to navigate the states in the session and see diffs representing the changes in each state. Of course, Magit can be used for a great many things, and you have that full power available to you to use with Mindstream sessions.

5.2 Git-Timemachine

The git-timemachine Emacs package is a great way to temporally navigate your session. Unlike the usual undo and redo operations which track edits with high granularity, mindstream sessions are bounded by `save-buffer` invocations which tend to represent natural, distinct stages in your development. Mindstream doesn't include a built-in way to navigate these states, but you can use the git-timemachine package to do this (in read-only mode).

5.3 Previewing

Quick feedback loops are the engines of creative progress. With this in mind, for whatever you're writing, it's valuable to have a way to preview what you've produced in output form. For instance, if you're writing documentation, you should have a keybinding to quickly build the file into HTML or a PDF, or render it within the buffer itself (as LaTeX modes sometimes allow), for you to review as you go. Likewise, if you're writing code, you should have a way to quickly evaluate the contents of your buffer and see the result.

This tip is not about Mindstream specifically but more about a good workflow to develop with the major mode you're using. For instance, with Racket Mode, it would be advisable to bind the command `racket-run` so that you can quickly see the output of your code. This command also saves the buffer so that the session history would represent natural points at which you felt the code was worth trying out. Similarly, if you're writing Markdown or reStructuredText, you should explore the features provided by the relevant major modes that would allow you to preview the produced documentation in HTML form with the right keybinding incantation.

5.4 Marking Significant Versions

Mindstream sessions can have *a lot* of commits, and they ensure that you never need to name your document at different stages of development out of fear that you'll lose important work. But if you happen to be writing a book, say, you might like to mark specific points as being significant in some way, so that if you ever have to search through earlier versions of your work, you'll know where to look. Traditionally, you might rename your file something like `draft_first_revision_final.tex`. From Mindstream's perspective, this a confusion of space and time. We seek to capture a

moment in time, but we use a distinct place in space (a new file) for this purpose. In Mindstream, it'd be better to use a standard feature of Git, *tagging*, to achieve the same result. That is, when you arrive at a version you think is significant, simply tag the current commit (using a Git client of your choice, such as Magit) with a name and a message. It's a much more useful way to do it than `draft_final_final_....tex`! For example, it allows you to very quickly switch to different "good" versions (e.g. in Magit, `bb` and then select a tag in the completion menu) that you could show to editors for proofreading.

5.5 Choosing an Archive Path

Mindstream stores anonymous sessions at `mindstream-path` (default: `~/mindstream/anon`) in a randomly generated folder name filed under the session creation date under the name of the template used to create the session. Sessions that you don't save are typically archived instead, which moves them to `mindstream-archive-path` (default: `~/mindstream/archive`) following the same organization scheme.

The default value of `mindstream-archive-path` is a safe and good choice. But you may like to do things differently, and in that case, here are some options to consider.

It's likely that you will work on dozens, hundreds, or thousands of anonymous sessions over time, of which you will only save a small minority. For the remaining, archived, sessions, though you may not wish to work on them further, they still serve as a detailed and organized log of your thoughts and work over time, and you may occasionally find a need to refer to them. For this reason, the archive is valuable and we recommend preserving this default behavior.

But if you prefer to consider anonymous sessions as ephemeral, to be forgotten if unsaved (perhaps you can only be truly creative if you can scrunch up that session and toss it in the bin so no one ever sees it again!), then you may prefer to delete the archive from time to time. Many operating systems provide standard ways to do this kind of thing – *temp folders*, usually named `tmp` – which are occasionally cleared automatically by the operating system, without requiring you to manage this. If your operating system provides a good option here, you may prefer to use it.

5.5.1 `/var/tmp`

`/var/tmp` is a standard path on Unix systems for holding temporary files. Unfortunately, *there is no accepted convention* on its handling. Some systems clear its contents rarely or never, while others clear its contents *on every reboot*. As a primary use for Mindstream is for you to have a reliable place to capture your thoughts with very low overhead, it's important that you should feel relatively secure that if your system were to crash, you would still be able to recover any (anonymous) Mindstream sessions you may have been in the middle of.

So if you'd like to use `/var/tmp`, first check the contents of this folder and refer to the documentation on your particular system to see how it handles this path. If that behavior is predictable enough for you (e.g. say the folder is cleared only on OS upgrades), then you can use it like this:

```
:custom
...
(mindstream-path "/var/tmp/mindstream")
```

5.5.2 Home/tmp

Another option that's similar to this one but more predictable is to define a new path in your home folder for this purpose (say `~/tmp`), that you are at liberty to periodically clear yourself, and which you could share across all applications for this purpose. If you go with this option, you can use this path in Mindstream like so:

```
:custom
...
(mindstream-path
 (concat (file-name-as-directory (getenv "HOME"))
         "tmp/mindstream"))
```

Remember that the path we are configuring here is for *anonymous sessions* only. If you decide to keep a session around and save it via `mindstream-save` (default binding: `C-c , C-s`), it would be saved to `mindstream-save-session-path` which defaults to `~/mindstream/saved`. You can customize this as well, of course:

```
:custom
...
(mindstream-save-session-path
 (concat (file-name-as-directory (getenv "HOME"))
         "my/mindstream/sessions/path"))
```

5.6 Organizing Sessions

Anonymous sessions, and consequently archived sessions, are filed under their creation date under the template used to create them, in a folder structure reflecting this scheme. For sessions saved by you to another location, by default, these are saved to `mindstream-save-session-path`, filed just under the template used to create them (and not by creation date).

This is a uniform and useful filing scheme. Even so, over time, you may want to rename some of your saved sessions at `mindstream-save-session-path`, or move them to a new location, or reorganize them in some other way. What special features does Mindstream provide for this? *None*! Remember, Mindstream sessions are just *ordinary folders* containing *ordinary Git repositories*. You can use normal Emacs or shell tools to rename, delete, organize them as you see fit, and you would still be able to load them in Mindstream as usual (once you navigate to what may be their new locations).

6 Acknowledgements

This package was conceived in discussion with Greg Hendershott.

”Live mode” was inspired by coding demos given by Matthew Flatt using DrRacket.

7 Non-Ownership

This work is not owned by anyone. Please see the Declaration of Non-Ownership.