Scripts I wrote that I use all the time

by Evan Hahn, posted Oct 22, 2025

In my decade-plus of maintaining my dotfiles, I've written a lot of little shell scripts. Here's a big list of my personal favorites.

Clipboard

copy and pasta are simple wrappers around system clipboard managers, like pbcopy on macOS and xclip on Linux. I use these *all the time*.

```
# High level examples
run_some_command | copy
pasta > file_from_my_clipboard.txt

# Copy a file's contents
copy < file.txt

# Open a file path from your clipboard
vim "$(pasta)"

# Decode some base64 from the clipboard
pasta | base64 --decode
```

pastas prints the current state of your clipboard to stdout, and then whenever the clipboard changes, it prints the new version. I use this once a week or so.

```
# High level example
pastas > everything_i_copied.txt

# Download every link I copy to my clipboard
pastas | wget -i -
```

cpwd copies the current directory to the clipboard. Basically `pwd | copy`. I often use this when I'm in a directory and I want use that directory in another terminal tab; I

copy it in one tab and `cd` to it in another. I use this once a day or so.

File management

`mkcd foo` makes a directory and cds inside. It's basically `mkdir foo && cd foo`. I use this *all the time*—almost every time I make a directory, I want to go in there.

`tempe` changes to a temporary directory. It's basically `cd "$(mktemp -d)"`. I use this *all the time* to hop into a sandbox directory. It saves me from having to manually clean up my work. A couple of common examples:

```
# Download a file and extract it
tempe
wget 'https://example.com/big_file.tar.xz'
tar -xf big_file.tar.xz
# ...do something with the file...

# Write a quick throwaway script to try something out
tempe
vim foo.py
python3 foo.py
```

`trash a.txt b.png` moves `a.txt` and `b.png` to the trash. Supports macOS and Linux. I use this every day. I definitely run it more than `rm`, and it saves me from accidentally deleting files.

`mksh` makes it quick to create shell scripts. `mksh foo.sh` creates `foo.sh`, makes it executable with `chmod u+x`, adds some nice Bash prefixes, and opens it with my editor (Vim in my case). I use this every few days. Many of the scripts in this post were made with this helper!

Internet

serveit starts a static file server on `localhost:8000` in the current directory. It's basically `python3 -m http.server 8000` but handles cases where Python isn't installed, falling back to other programs. I use this a few times a week. Probably less useful if you're not a web developer.

getsong uses `yt-dlp` to download songs, often from YouTube or SoundCloud, in the highest available quality. For example, `getsong https://www.youtube.com/watch?v=dQw4w9WgXcQ` downloads that video as a song. I use this a few times a week… typically to grab video game soundtracks…

getpod similarly uses `yt-dlp` to download something for a podcast player. There are a lot of videos that I'd rather listen to like a podcast. I use this a few times a month.

getsubs downloads the English subtitles for a video. (There's some fanciness to look for "official" subtitles, falling back to auto-generated subtitles.) Sometimes I read the subtitles manually, sometimes I run `getsubs https://video.example/foo | ollama run llama3.2 "Summarize this"`, sometimes I just want it as a backup of a video I don't want to save on my computer. I use this every few days.

`wifi off`, `wifi on`, and `wifi toggle` are useful for controlling my system's wifi. `wifi toggle` is the one I use most often, when I'm having network trouble. I use this about once a month.

`url "$my_url"` parses a URL into its parts. I use this about once a month to pull data out of a URL, often because I don't want to click a nasty tracking link.

```
url 'https://evil.example/track-user-link?url=https%3A%2F%2Furl-i-want-to-visit.exampl
# original: https://evil.example/track-user-link?url=https%3A%2F%2Furl-i-want-to-visit
# protocol: https
# hostname: evil.example
# path: /track-user-link
# query: url=https%3A%2F%2Furl-i-want-to-visit.example&track=06f8582a-91e6-4c9c-bf8e-5
# - url https://url-i-want-to-visit.example
# - track 06f8582a-91e6-4c9c-bf8e-516884584aba
# hash: cookie=123
```

Text processing

`line 10` prints line 10 from stdin. For example, `cat some_big_file | line 10` prints line 10 of a file. This feels like one of those things that should be built in, like head and `tail`. I use this about once a month.

`scratch` opens a temporary Vim buffer. It's basically an alias for `$EDITOR $(mktemp)`. I use this about once a day for quick text manipulation tasks, or to take a little throwaway note.

`straightquote` converts "smart quotes" to "straight quotes" (sometimes called "dumb quotes"). I don't care much about these in general, but they sometimes weasel their way into code I'm working on. It *can* also make the file size smaller, which is occasionally useful. I use this at least once a week.

`markdownquote` adds > before every line. I use it in Vim a lot; I select a region and then run `:'<,'>!markdownquote` to quote the selection. I use this about once a week.

`length foo` returns 3. (I should probably just use `wc -c`.)

`jsonformat` takes JSON at stdin and pretty-prints it to stdout. I use this a few times a year.

`uppered` and `lowered` convert strings to upper and lowercase. For example, `echo foo | uppered` returns FOO. I use these about once a week.

`nato bar` returns `Bravo Alfa Romeo`. I use this most often when talking to customer service and need to read out a long alphanumeric string, which has only happened a couple of times in my whole life. But it's sometimes useful!

`u+ 2025` returns `ñ, LATIN SMALL LETTER N WITH TILDE`. A quick way to do a lookup of a Unicode string. I don't use this one *that* often…probably about once a month.

`snippets foo` cats `~/.config/evanhahn-snippets/foo`. I use `snippet arrow` for →, `snippet recruiter` for a quick "not interested" response to job recruiters, `snippet lorem` to print a "Lorem ipsum" block, and a few others. I probably use one or two of these a week.

REPL launchers

Inspired by Ruby's built-in `irb` REPL, I've made:

- `iclj` to start a Clojure REPL
- `ijs` to start a Deno REPL (or a Node REPL when Deno is missing)
- `iphp` to start a PHP REPL
- `ipy` to start a Python REPL
- `isql` to start a SQLite shell (an alias for `sqlite3 :memory:`)

Dates and times

`hoy` prints the current date in ISO format, like `2020-04-20`. I use this *all the time* because I like to prefix files with the current date.

`timer 10m` starts a timer for 10 minutes, then (1) plays an audible ring sound (2) sends an OS notification (see `notify` below). I often use `bb timer 5m` to start a 5 minute timer in the background (see `bb` below). I use this almost every day as a useful way to keep on track of time.

`rn` prints the current time and date using `date` and `cal`. I probably use it once a week. It prints something like this:

```
4:20PM on Wednesday, October 22, 2025

   September 2025
```

```
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30
```

Audio and video and pictures

`ocr my_image.png` extracts text from an image and prints it to stdout. It only works on macOS, unfortunately, but I want to fix that. (I wrote a post about this script.)

`boop` (an alias, not a shell script) makes a happy sound if the previous command succeeded and a sad sound otherwise. I do things like `run_the_tests ; boop` which will tell me, audibly, whether the tests succeed. It's also helpful for long-running commands, because you get a little alert when they're done. I use this *all the time*.

`sfx foo` basically just plays `~/.config/evanhahn-sfx/foo.ogg`. Used in `boop` and `timer` above.

`tunes` uses `mpv` to play audio from a file. I use this *all the time*, running `tunes --shuffle ~/music`.

`pix` uses `mpv` to show a picture. I use this a few times a week to look at photos.

`radio` is a little wrapper around some of my favorite internet radio stations. `radio lofi` and `radio salsa` are two of my favorites. I use this a few times a month.

`speak` reads from stdin, removes all Markdown formatting, and pipes it to a text-to-speech system (`say` on macOS and `espeak-ng` on Linux). I like using text-to-speech when I can't proofread out loud. I use this a few times a month.

`shrinkvid` is an `ffmpeg` wrapper that compresses a video a bit. I use this about once a month.

`removeexif` removes EXIF data from JPEGs. I don't use this much, in part because it doesn't remove EXIF data from *other* file formats like PNGs…but I keep it around because I hope to expand this one day.

`tuivid` is one I almost never use, but you can use it to watch videos in the terminal. It's cursed and I love it, even if I never use it.

Process management

`each` is my answer to xargs and `find ... -exec`, which I find hard to use. For example, `ls | each 'du -h {}'` runs `du -h` on every file in a directory. I use this infrequently but I always mess up xargs so this is a nice alternative.

`running foo` is like `ps aux | grep foo` but much easier (for me) to read—just the PID (highlighted in purple) and the command.

`murder foo` or `murder 1234` is a wrapper around `kill` that sends `kill -15 $PID`, waits a little, then sends `kill -2`, waits and sends `kill -1`, waits before finally sending `kill -9`. If I want a program to stop, I want to ask it nicely before getting more aggressive. I use this a few times a month.

`waitfor $PID` waits for a PID to exit before continuing. It also keeps the system from going to sleep. I use this about once a month to do things like:

```
# I want to start something only after another process finishes
waitfor 1234 ; something_else

# I started a long-running process and want to know when it's done
waitfor 1234 ; notify 'process 1234 is done'
```

`bb my_command` is like `my_command &` but it *really really* runs it in the background. You'll never hear from that program again. It's useful when you want to start a daemon or

long-running process you truly don't care about. I use `bb ollama serve` and `bb timer 5m` most often. I use this about once a day.

prettypath prints `$PATH` but with newlines separating entries, which makes it much easier to read. I use this pretty rarely—mostly just when I'm debugging a `$PATH` issue, which is unusual—but I'm glad I have it when I do.

`tryna my_command` runs `my_command` until it succeeds. `trynafail my_command` runs `my_command` until it fails. I don't use this much, but it's useful for various things. `tryna wget ...` will keep trying to download something. `trynafail npm test` will stop once my tests start failing.

Quick references

`emoji` is my emoji lookup helper. For example, `emoji cool` prints the following:

`httpstatus` prints all HTTP statuses. `httpstatus 204` prints `204 No Content`. As a web developer, I use this a few times a month, instead of looking it up online.

`alphabet` just prints the English alphabet in upper and lowercase. I use this surprisingly often (probably about once a month). It literally just prints this:

```
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

System management

`theme 0` changes my whole system to dark mode. `theme 1` changes it to light mode. It doesn't just change the OS theme—it also changes my Vim, Tmux, and terminal themes. I use this at least once a day.

`sleepybear` puts my system to sleep, and works on macOS and Linux. I use this a few times a week.

`ds-destroy` recursively deletes all `.DS_Store` files in a directory. I hate that macOS clutters directories with these files! I don't use this often, but I'm glad I have it when I need it.

Grab bag

`catbin foo` is basically `cat "$(which foo)"`. Useful for seeing the source code of a file in your path (used it for writing up this post, for example!). I use this a few times a month.

`notify` sends an OS notification. It's used in several of my other scripts (see above). I also do something like this about once a month:

```
run_some_long_running_process ; notify 'all done'
```

`uuid` prints a v4 UUID. I use this about once a month.

What about your scripts?

These are just scripts *I* use a lot. I hope some of them are useful to you!

If you liked this post, you might like “Why ‘alias’ is my last resort for aliases” and “A decade of dotfiles”.

Oh, and contact me if you have any scripts you think I’d like.