Modern CLI Replacements for the Shell Layer

Ronald G. Thomas

2026-04-27

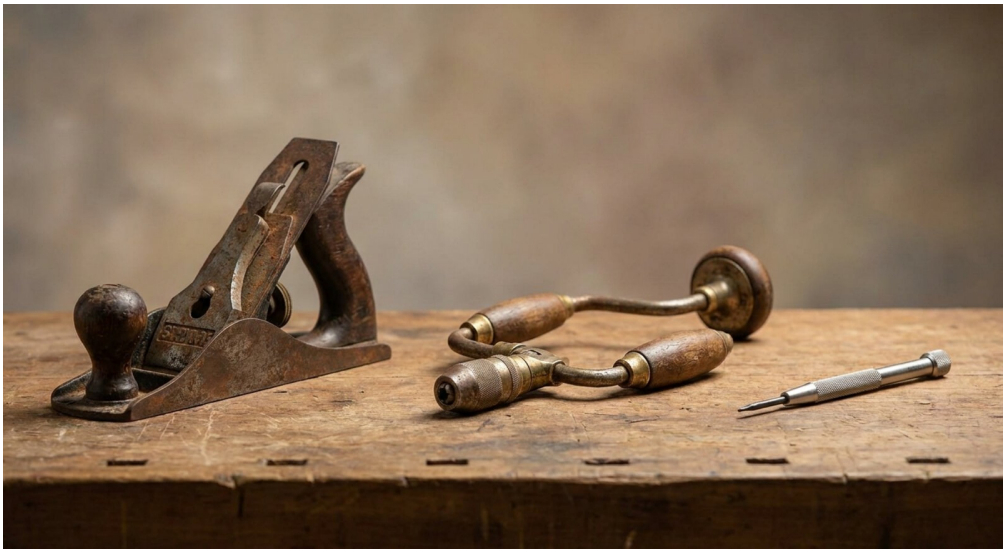

Figure 1: A bench of polished hand tools laid out in order of use, with each tool noticeably newer in design than its predecessor on the bench. The image evokes the idea that the same task (cutting, smoothing, fastening) can be accomplished with successive generations of tooling whose ergonomic differences accumulate into a meaningful change in how the work feels, even though the underlying purpose is unchanged.

Replacing the historical Unix utilities with their Rust-implemented counterparts is a Shell-layer extension that preserves the layer's contract while substantially improving its defaults.

Introduction

The historical Unix utilities (`grep`, `find`, `cat`, `ls`, `cd`, `git`) are, in a meaningful sense, finished software. Their interfaces have not changed in decades, their behavior is portable across every system likely to host them, and their performance is adequate for the cases they were originally designed to address. They are also, increasingly, tools whose defaults reflect the constraints of the environment that produced them rather than the environment in which they are now most commonly used. `grep` does not recursively search by default because in 1973 a recursive search across a single user's home directory was a non-trivial fraction of available compute. In 2026, recursing across a half-million-file

checkout is a sub-second operation, and the default has not caught up. Similarly, `find` does not respect `.gitignore` because `.gitignore` did not exist. `ls` does not show git status because git did not exist. `cd` does not learn frequently-visited directories because the disks of the era could not afford a frecency database.

A small set of Rust-implemented utilities, authored predominantly between 2017 and 2023, address these defaults without changing the interfaces that downstream scripts and muscle memory depend on. They are not faster reimplementations in the sense of producing identical output more quickly. They produce different output, by design, when the historical default is the part that the user most often wanted to override. The collection covered in this post (`ripgrep`, `fd`, `bat`, `eza`, `zoxide`, `delta`, `lazygit`, plus `fzf` as the connective tissue) is the most-installed subset and the one whose ergonomic gains are large enough to justify the configuration work.

More formally, we document here an extension to the Shell layer (Layer 4) of the Workflow Construct described in A Workflow Construct for the Modern Data Scientist. The Shell layer is substitutable (zsh can be replaced with bash, fish, or nushell), and so are the utilities the shell calls on. This post is the substitution catalog for the `grep` / `find` / `cat` / `ls` / `cd` / `git` family at the layer below the shell itself, with explicit migration notes for the autojump plugin already present in the construct.

Motivations

The pain points that motivated documenting these tools as a construct extension are specific:

- The `grep` reflex of `grep -r` plus a shell-quoted glob pattern produces matches inside `node_modules/`, `.git/`, build directories, and binary files in roughly equal measure. The signal-to-noise ratio degrades to the point where the output is more frustrating than absent.
- The `find` syntax (`find . -name '*.R' -not -path './renv/*'`) is hard to remember between sessions and hard to read in a script. The cost of forgetting the syntax is a context switch to read the man page; the cost of remembering it imperfectly is silent inclusion or exclusion of files the user did not intend.
- The `cd` reflex (`cd ~/Dropbox/prj/03-name/...`) is re-typed dozens of times per day across the same set of project roots. Tab completion helps but does not learn.
- `git diff` in a terminal that does not paginate syntax-highlighted output is the operation the author most often defers to RStudio for, despite the rest of their workflow being terminal-native. The friction is small per invocation and large in aggregate.
- The construct already includes `j` (autojump) for fuzzy directory navigation. The autojump implementation (Python, single user-level database, occasional startup-time slowness on large `cdpath` sets) has been superseded by `zoxide` (Rust, multi-user-aware, faster startup), and the migration preserves the `j` muscle memory.

Objectives

The post sets out to deliver:

1. Installation commands for each tool on macOS (Homebrew) and Debian-derived Linux (`apt` plus the project's GitHub releases when the upstream package lags).
2. A single canonical `.zshrc` block that activates each tool, with the autojump-to-zoxide migration handled in-place rather than as a separate step.

3. A daily-workflow command table mapping each historical utility to its replacement, with the substitution verified for the most common invocations the author actually types.
4. A failure-mode catalog (Things to Watch Out For) covering the cases in which the new defaults produce surprising results.

The reader should be able to install, configure, and verify the full set in approximately one hour, after which the historical utilities can be retained as fallbacks (they are not removed) but used principally for portability work and remote sessions where the modern tools are absent.

Figure 2: A magnifying glass resting on an open book page, focused on a single highlighted line, on a wooden desk.

What Is the Modern CLI Replacement Family?

The eight tools covered here are not a coordinated distribution; each was authored independently by different maintainers and adopts its own configuration conventions. What unifies them is a set of design choices that each project arrived at without coordination:

- **Sensible defaults.** Recursion, colorization, and respect for `.gitignore` are on by default. The user should not have to remember three flags to make the tool do the obviously-useful thing.
- **Performance through Rust.** The tools are implemented in a language that produces binaries with predictable performance characteristics on large inputs. The difference matters most on directory trees with hundreds of thousands of files; on smaller inputs the gain is imperceptible but the defaults still differ.
- **Composability with `fzf`.** Each tool produces output that pipes cleanly into a fuzzy finder for interactive selection. The composition replaces several historical patterns (interactive command-history substitution, file selection, process killing) with a single muscle memory.

- **Optional substitution.** Each tool can be aliased over the historical utility (`alias grep='rg'`) or kept under its own name. The construct documented here keeps both available: the modern tool under its native name for interactive use, the historical tool unchanged for scripts and remote sessions.

The collection is not exhaustive. Notable adjacent tools that are deliberately not covered include `procs` (replacement for `ps`), `dust` (replacement for `du`), `bottom` (replacement for `top`), and `sd` (replacement for `sed`). They follow the same design pattern; the case for adopting them is real but is not load-bearing for the construct, so they are left as a later optional extension.

Prerequisites

The configuration assumes:

- **Operating system:** macOS 13+ with Homebrew, or Debian-derived Linux with `apt` and access to the project's GitHub releases for the few cases where the distribution package lags upstream.
- **Shell:** zsh 5.0 or later (post 01 prerequisite). The configuration block below is zsh-specific; bash users will need to translate the function syntax.
- **Existing construct setup:** the dotfiles repository (post 24) and the zsh setup (post 01) should already be in place. The configuration here is a single block added to `.zshrc`; the canonical place for it is just below the existing FZF configuration.
- **Time investment:** approximately 30 minutes for installation, 15 minutes for configuration, and a further 15 minutes for the autojump-to-zoxide database migration if applicable. The total is under an hour for users starting from the construct's existing zsh setup.

Installation

macOS (Homebrew)

```
brew install ripgrep fd bat eza zoxide git-delta lazygit fzf

# Verify each is on $PATH
for t in rg fd bat eza zoxide delta lazygit fzf; do
  command -v "$t" > /dev/null && echo "$t: $(command -v $t)" \
    || echo "$t: MISSING"
done
```

Debian-derived Linux (Ubuntu 24.04, Mint 22)

The Debian archive lags upstream for several of these tools. The recipe below uses `apt` where the version is acceptable and the upstream release otherwise.

```
sudo apt update
sudo apt install -y ripgrep fd-find bat fzf

# fd is named fdfind on Debian to avoid conflict with an older tool;
# install a personal symlink to fd
mkdir -p ~/.local/bin
ln -sf "$(command -v fdfind)" ~/.local/bin/fd

# bat is named batcat on Debian for the same reason
ln -sf "$(command -v batcat)" ~/.local/bin/bat

# eza, zoxide, delta, lazygit are not in the Debian archive (or lag)
# Install from upstream releases:
EZA_VERSION=$(curl -s https://api.github.com/repos/eza-community/eza/releases/latest | jq -r .tag_name)
# (continue per upstream's instructions)
```

The Debian Linux installation is consciously more hand-rolled than the macOS path; this is the one place where the polyglot shell scripts module of the construct earns its keep.

Configuration

The full `.zshrc` block to activate the family follows. It is intended to live just below the existing FZF block in `~/Dropbox/dotfiles/zshrc` (the construct's canonical dotfile location, post 24).

```
# ============================================================
# Modern CLI Replacements (post 53)
# ============================================================

# zoxide: replaces cd's frequency table; '--cmd j' aliases the
# binary to 'j', preserving the autojump muscle memory while
# migrating the database.
if command -v zoxide > /dev/null 2>&1; then
  eval "$(zoxide init zsh --cmd j)"
fi

# eza: ls replacement with git awareness and tree mode
if command -v eza > /dev/null 2>&1; then
  alias ls='eza --group-directories-first'
  alias ll='eza -l --git --group-directories-first'
  alias la='eza -la --git --group-directories-first'
  alias lt='eza --tree --level=2 --git-ignore'
fi

# bat: cat with syntax highlighting; only alias for interactive
# use, because pipelines that consume cat's output assume
# unstyled bytes.
```

```
if command -v bat > /dev/null 2>&1; then
  alias cat='bat --paging=never --style=plain'
  # Force plain output when piping (defensive)
  export BAT_PAGER='less -R'
fi

# delta: pager for git diff
if command -v delta > /dev/null 2>&1; then
  export GIT_PAGER='delta'
fi

# fd, ripgrep: no aliasing required (their names are distinct).
# ripgrep is already configured as FZF_DEFAULT_COMMAND in the
# zshrc's FZF block.

# Optional: lazygit alias (not auto-aliased to git so that
# scripted git commands still hit the porcelain).
if command -v lazygit > /dev/null 2>&1; then
  alias lg='lazygit'
fi
```

The `git config` for delta requires a separate edit, since it lives in `~/.gitconfig` rather than `.zshrc`:

```
[core]
    pager = delta
[interactive]
    diffFilter = delta --color-only
[delta]
    navigate = true
    light = false
    line-numbers = true
    side-by-side = true
[merge]
    conflictstyle = diff3
[diff]
    colorMoved = default
```

The `delta` configuration is presented as a complete file fragment because partial fragments are the source of the most frequent reader confusion (the `[merge]` and `[diff]` sections are required for `delta`'s side-by-side merge display, and are easy to omit).

Migration: autojump to zoxide

The construct currently activates autojump via the plugin-loader block in `.zshrc` (the `[[ -s $BREW_PREFIX/etc/profile.d/autojump.sh ]] && source ...` line). Migrating to zoxide while preserving the `j` muscle memory is a three-step substitution:

1. Add the `zoxide init zsh --cmd j` line shown above. With both autojump and zoxide active, `j` resolves to whichever was sourced last; the order matters.
2. Comment out the autojump source line. Zoxide is now exclusive on the `j` keyword.
3. Optionally seed the zoxide database with the autojump history:

   ```
   awk '{print $2}' ~/.local/share/autojump/autojump.txt | \
     while read -r dir; do zoxide add "$dir"; done
   ```

   This preserves the user's accumulated frecency data rather than starting fresh.

After a few days of use, the autojump installation can be fully removed:

```
brew uninstall autojump   # macOS
# or apt remove autojump  # Debian
rm -rf ~/.local/share/autojump
```

Verification

The block below confirms each tool is installed, on `$PATH`, and reachable through its expected name. Run after sourcing the new `.zshrc`.

```
# Each command should print a version line; failures localise
# to the corresponding tool.
rg --version | head -1
fd --version
bat --version
eza --version
zoxide --version
delta --version
lazygit --version
fzf --version

# Functional smoke tests
rg 'ggplot' --type r .                    # Should respect .gitignore
fd '\.qmd$'                                # Should match recursively
bat README.md | head -5                    # Should syntax-highlight
eza --tree --level=2 --git-ignore         # Should show git-aware listing
j workflow                                 # Should jump (zoxide)
git diff HEAD~1 HEAD                       # Should display via delta
```

A green pass on all eight version commands plus a working `j` jump indicates the migration is complete.

Daily Workflow

Once configured, the daily-use substitution table is small and stable. The historical utilities remain available for edge cases where the modern defaults are wrong.

Task	Historical	Modern	Reason for the modern default
Recursive text search	`grep -r 'pattern' .`	`rg 'pattern'`	Recursive by default, respects `.gitignore`, parallel
Find files by name	`find . -name '*.R'`	`fd '\.R$'`	Shorter syntax, parallel, respects `.gitignore`
Display file with highlighting	`cat file.R | less`	`bat file.R`	Syntax highlighting, line numbers
List with git status	`ls -la`	`eza -la --git`	Git-aware columns, more readable defaults
List as tree	`find . -type d | head`	`eza --tree --level=2`	Purpose-built
Jump to known directory	`cd ~/Dropbox/prj/...`	`j keyword`	Frecency-based, fuzzy
View git diff	`git diff`	`git diff` (with delta pager)	Syntax-highlighted, hunk-anchored
Interactive git review	`git log; git show ...`	`lazygit`	Terminal UI
Interactive selection	`Ctrl-r` (history)	`Ctrl-r` (fzf)	Fuzzy across history, files, processes

The substitution is partial by design: scripts continue to use the historical names (`grep`, `find`) for portability, and the modern tools are invoked under their native names (`rg`, `fd`) for interactive use. Aliasing `grep` to `rg` globally is possible but breaks scripts that rely on `grep`'s exact output format; the aliasing here is limited to `ls`, `cat`, and `git diff` (via `GIT_PAGER`), which are display-only commands whose output is not parsed by other tools.

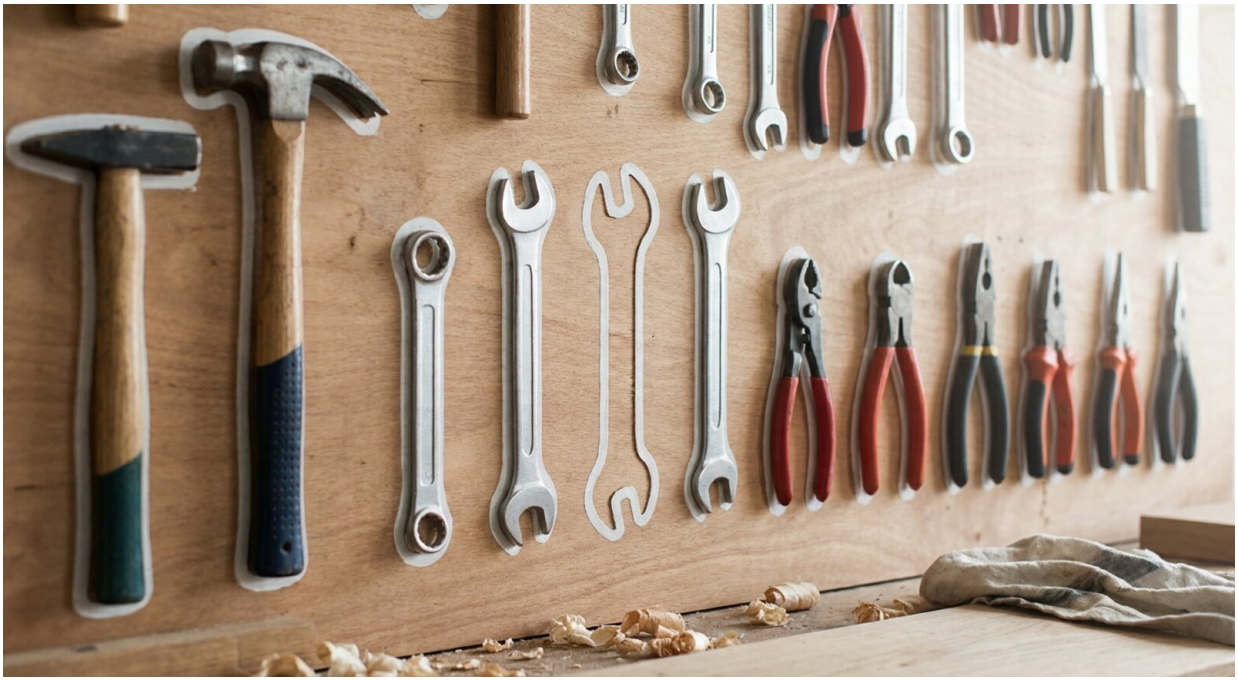

Figure 3: A well-organized pegboard wall in a workshop, tools hung in labeled silhouette outlines, one outline left empty.

Things to Watch Out For

Six gotchas have surfaced repeatedly during real use of this configuration. Each is small in isolation; in aggregation they are the most common reasons users abandon the migration in the first week.

- **`ripgrep` ignores `.gitignore` by default; this is occasionally undesirable.** Add `--no-ignore` to search inside ignored directories (for example, when grepping through `node_modules/` for a third-party bug). The flag is the most common one to need; aliasing `rg='rg --no-ignore-vcs'` is over-aggressive and not recommended.
- **`bat` aliased over `cat` breaks pipelines that consume literal bytes.** The configuration above sets `--paging=never --style=plain` and `BAT_PAGER` to mitigate, but pipelines that depend on `cat`'s exact byte-for-byte output (uncommon but real, e.g., when passing a file to a hash function) should call `\cat` (with backslash) or `/bin/cat` to bypass the alias.
- **`eza` icons require a Nerd Font.** The default `eza` output is fine without icons; adding `--icons` produces glyph rendering that requires a patched font (Hack Nerd Font, FiraCode Nerd Font, etc.). Without the font, the icons render as Unicode replacement squares. Either install a Nerd Font in the terminal or omit `--icons`.
- **`lazygit` and `delta` interact in a non-obvious way.** `lazygit` has its own diff viewer that does not call `GIT_PAGER`; setting `delta` as the pager affects `git diff` from the shell but not `lazygit`'s internal diff display. `lazygit` can be configured to use `delta` via `~/.config/lazygit/config.yml` (`gui.pagerPath: delta`) if uniform diff styling is desired.
- **`zoxide`'s `j` does not include directories that have not been visited via the underlying `cd` at least once while zoxide is installed.** A new clone is invisible to `j` until the user

`cd`s into it manually the first time. The autojump-to-zoxide migration recipe above seeds the database to mitigate.

- **`fd`'s default ignores hidden files (those starting with `.`).** `fd 'config'` will not match `.gitignore`, `.zshrc`, or `.config/...`. Use `fd -H 'config'` to include hidden entries; the equivalent `find` reflex matches them by default, which is the opposite of `fd`'s default and the second most common cause of confusion in the first week.
- **macOS `find -regex` and `fd '...'` use different regex syntaxes (BSD vs. Rust).** Patterns translated literally fail in unobvious ways. The `fd` documentation lists the correspondences; a copy is worth keeping nearby until the Rust regex syntax is internalized.

Uninstall / Rollback

The migration is non-destructive. Each tool can be removed without affecting the others; removing all of them returns the shell to its pre-installation state.

```
# Remove tools (macOS)
brew uninstall ripgrep fd bat eza zoxide git-delta lazygit fzf

# Remove the .zshrc block: delete or comment out the
# 'Modern CLI Replacements (post 53)' section.

# Remove delta from ~/.gitconfig: delete the [core], [delta],
# [merge], [diff], [interactive] sections (or revert the
# values to git's defaults).

# Remove zoxide's database (optional)
rm -rf ~/.local/share/zoxide

# Restore autojump if the migration was reversed
brew install autojump
# Add the autojump source line back to .zshrc
```

The historical utilities (`grep`, `find`, `cat`, `ls`, `cd`) are unaffected by any of the above; they remain available at their canonical paths regardless of which modern tools are installed.

Figure 4: A row of vintage and modern stopwatches lined up on a wooden table, the modern one at the end mid-click with a faint motion blur.

Lessons Learned

Working through this migration on three machines (the construct's MacBook, the ThinkPad with Linux Mint, and a clean EC2 Ubuntu instance) surfaced lessons grouped into three buckets.

Conceptual

- **A Shell-layer extension is exactly that, an extension.** None of the tools above replaces the layer; each adds an alternative under a new name and an optional alias under the historical name. This is the cleanest pattern for any layer-extension work in the construct: keep the historical tool reachable, add the modern tool alongside, let the user choose per invocation.
- **Sensible defaults are the bulk of the value, not raw speed.** Of the eight tools, only `ripgrep` and `fd` show meaningfully better wall-clock performance on small inputs. The other six are roughly comparable in speed to their historical predecessors and earn their place through colorization, git awareness, syntax highlighting, and `.gitignore` respect. Speed matters on large inputs but is not the primary reason to adopt.
- **Frecency-based directory navigation pays the largest per-keystroke dividend.** The substitution from `cd ~/Dropbox/prj/03-name/...` to `j 03` saves a measurable fraction of a working day across hundreds of invocations. None of the other substitutions has the same per-keystroke ratio.
- **Polyglot shell-script modules earn their keep on the Linux side.** The macOS path is uniformly `brew install`; the Debian path is a mix of `apt install`, project release downloads,

and personal `~/.local/bin` symlinks. The construct's shell-scripts row absorbs this complexity by version-controlling the install recipe rather than re-deriving it on each new machine.

Technical

- **`fzf` is the connective tissue, not a peer.** The other seven tools are individually useful; the composition becomes meaningfully more powerful when piped into `fzf` for interactive selection. The construct's existing `FZF_DEFAULT_COMMAND='rg --files --hidden'` line is the keystone of this composition.
- **`zoxide`'s `--cmd j` flag is the migration's hinge.** Without it, the migration would require relearning a new command (`z` instead of `j`) and the muscle memory cost would dominate the ergonomic gain. With it, the migration is invisible to the user's daily flow.
- **`delta`'s side-by-side mode requires sufficient terminal width.** A laptop screen at default font size has ~80 columns; side-by-side becomes unreadable below ~120. Either set a smaller font for git operations or disable `side-by-side` for narrow terminals via a conditional `~/.gitconfig` include.
- **`bat`'s syntax detection occasionally guesses wrong on ambiguous extensions.** `.R` is correctly inferred as R; `.Rmd` is sometimes inferred as Markdown only (without the R-chunk highlighting). Force the language with `bat --language=Rmd file.Rmd` when needed.

Gotcha-shaped

- **`eza` does not have a stable command-line flag set across major versions.** A `.zshrc` alias that worked with eza 0.18 may need a flag swap on 0.20 (the `--git-ignore` flag changed semantics). Pin a minimum version in the version matrix below and re-verify aliases after an upgrade.
- **`git-delta` is the Homebrew package name; the binary is `delta`. Do not be confused by the apparent mismatch.** On Linux, the upstream tarball name is `delta-${VERSION}-x86_64-unknown-linux-musl.tar.gz`, which produces a `delta` binary that should be moved to `~/.local/bin/`.
- **`fd`'s `-name` semantic is matching, not exclusion.** `fd -name foo` finds files named `foo`; the analogous `find . -name 'foo'` is the same thing but super-cilious users sometimes type `fd -not foo` expecting an exclusion (the correct flag is `--exclude foo`).

Limitations

The migration as documented has the following honest limitations:

- **It is opinionated about zsh.** The configuration block is zsh-specific. Bash users can adapt the alias and `eval` lines; fish users will need fish-native equivalents (zoxide ships them upstream).
- **The Debian installation is more involved than macOS.** The `brew install` one-liner has no Linux equivalent because the upstream releases lag the Debian archive for several tools. The construct's shell-scripts row should absorb the Debian recipe; until it does, the Linux user carries the recipe in this post.

- **The historical utilities are not removed.** The configuration leaves `grep`, `find`, `cat`, `ls`, and `cd` in place. This is deliberate (scripts and remote sessions need them) but it means the user does not uninstall anything; the disk and `$PATH` footprint of the workstation grows by approximately 50 MB across the eight tools.
- **`fzf` is required for `Ctrl-r` history search but is not installed by `brew install ripgrep` etc.** It must be installed separately. The post recipe includes it, but a user copying only the `ripgrep` line from a collaborator's snippet may miss the dependency.
- **None of these tools addresses regex differences across systems.** A pattern that matches in `ripgrep` may not match in `grep -E` and may not match in `awk`. The modern tools standardize on Rust regex, which is itself a dialect; portability across the three regex worlds (POSIX, PCRE, Rust) remains a per-tool concern.

Opportunities for Improvement

Several extensions are plausible for subsequent revisions of this post or related construct work:

1. **A second pass for `procs`, `dust`, and `bottom`.** The `ps` / `du` / `top` triumvirate has Rust replacements that follow the same pattern. They are not load-bearing for the construct but are pleasant to have; a follow-up post could document them as a Tier-A optional addition.
2. **A Nerd Font setup post.** The icon-rendering prerequisite for `eza --icons` is a small but persistent blocker; a one-page post documenting font selection, installation, and terminal configuration would close the gap.
3. **`lazygit` configuration walkthrough.** The default `lazygit` is fine; the configurable bits (delta as diff viewer, custom command bar shortcuts, alternate keybindings for `vim`-style users) would warrant a companion post if the reader spends substantial time in the TUI.
4. **A `mise`-managed installation path.** `mise` (covered in post 54) can pin the versions of these binaries for per-project reproducibility. The version-pinning approach is overkill for personal use but is the right layer for team-scale installations where bug reports tracing to subtle eza / delta version differences between collaborators are common.
5. **A Linux-side install script in `~/bin/`.** The Debian recipe in this post should be promoted to a first-class shell script in the construct's shell-scripts row. The script accepts a list of tool names and installs them via `apt`, `~/.local/bin` symlink, or upstream release as appropriate.

Wrapping Up

The historical Unix utilities are not broken; they are simply optimized for a computing environment that no longer predominates. Their recursive search is opt-in, their tree listing is hand-rolled, their colorization is bolted on, and their git awareness is non-existent, all because those features were not in scope when they were written. The Rust-implemented replacements documented here invert each of those defaults without changing the layer's contract: the shell still calls a binary, the binary still emits text on stdout, and downstream pipelines still receive bytes they can parse.

The migration is cheap (under an hour for the full set) and non-destructive (the historical tools remain available under their canonical names). The largest single ergonomic gain comes from `zoxide`-via-`j`, which preserves the construct's existing autojump muscle memory while upgrading the

underlying frecency database; the second largest comes from `ripgrep`'s default `.gitignore` respect. The remaining tools earn their place individually but contribute less to the per-keystroke ratio.

The migration's principal failure mode is not technical but attentional: a user who installs the tools without working through the gotcha catalog (Things to Watch Out For above) will encounter several surprises in the first week that look like tool bugs and are actually intentional default differences. The catalog exists for that reason.

In conclusion, four points merit emphasis. First, modern CLI replacements are a Shell-layer extension, not a substitute: they sit alongside the historical tools rather than replacing them. Second, the largest ergonomic gain is per-keystroke rather than per-second; sensible defaults (`.gitignore` respect, recursive search, frecency navigation) accumulate value faster than raw performance. Third, the autojump-to-zoxide migration is the single most consequential change: `zoxide init zsh --cmd j` preserves the muscle memory and upgrades the underlying database in one line. Fourth, the gotchas should be read before installing, not after; the first-week surprises are predictable and the time to read about them is much shorter than the time to debug them.

See Also

- Posts in this repository that document adjacent layers:
  - Unix Command-Line Workspace Setup: the Shell layer setup that this post extends.
  - Multi-Laptop macOS Bootstrap: the workstation-IaC keystone where the configuration block above lives.
  - The `zzgit` secret-scanning git wizard that pairs with `lazygit` for a complete git-side workflow.
  - Refactoring a Personal Toolbox: Scripts versus Shell Functions: the distinction between shell scripts and functions, which governs whether the install recipe in this post should live in a `~/bin/` script or a `.zshrc` function.
  - A Workflow Construct for the Modern Data Scientist: the construct framing that names the Shell-layer extension to which this post belongs.
- Project repositories for each tool:
  - `ripgrep`: https://github.com/BurntSushi/ripgrep
  - `fd`: https://github.com/sharkdp/fd
  - `bat`: https://github.com/sharkdp/bat
  - `eza`: https://github.com/eza-community/eza
  - `zoxide`: https://github.com/ajeetdsouza/zoxide
  - `delta`: https://github.com/dandavison/delta
  - `lazygit`: https://github.com/jesseduffield/lazygit
  - `fzf`: https://github.com/junegunn/fzf

Reproducibility

The configuration was developed and verified on the following software stack:

Component	Version	Notes
Operating system	macOS 15 (Sequoia)	primary daily driver
Operating system	Linux Mint 22 (Wilma)	secondary verification
Operating system	Ubuntu 24.04 LTS (EC2)	clean-room verification
Shell	zsh 5.9	minimum 5.0
`ripgrep`	14.1	minimum 13
`fd`	9.0	minimum 8.7
`bat`	0.24	minimum 0.22
`eza`	0.20	flag set stable since 0.20
`zoxide`	0.9.4	`--cmd j` available since 0.6
`delta`	0.18	minimum 0.16 for current config keys
`lazygit`	0.44	minimum 0.40
`fzf`	0.46	minimum 0.30
Homebrew	4.4 (macOS)	for the macOS install path

Date of last verification: 2026-04-27.

Feedback

Corrections, suggestions, and questions are welcome. Please open an issue or pull request on the GitHub repository or send an email to user@example.com. Substitutions for any single tool are particularly welcome (e.g., `helix-editor`'s built-in fuzzy finder as an alternative to `fzf`-plus-the-collection); they will be incorporated into a subsequent revision of this post.

Related posts in this cluster

This post is part of the *Workflow Construct* series. Recommended reading order:

1. Post 15: A Workflow Construct for the Modern Data Scientist
2. Post 16: Unix Command-Line Workspace Setup for Data Science
3. Post 17: Multi-Laptop macOS Bootstrap
4. Post 18: Setting Up Git for Data Science Workflows
5. Post 19: Setting Up Neovim as a Data Science IDE
6. **Post 21: Modern CLI Replacements for the Shell Layer** (this post)
7. Post 25: Install Linux Mint on a MacBook Air