Here are my favorite hacks

- I find bash "ctrl+R" history search command to be a "hard movement" for my fingers, so I have the following bindings in my bash setup scripts:

  ```
  # Do these only if it's a proper tty. This way we skip them for ssh logins that
  # execute a command (without a proper tty), such as scp, and would print a
  warning/error
  # (causing nonzero return code which a host script could treat as failure)
  if [[ "$-" =~ "i" ]]; then
    # Cool bindings to get `ctrl-R' behavior simply with up/down arrows
    bind '"\e[A": history-search-backward'
    bind '"\e[B": history-search-forward'
  fi
  ```
- I have several aliases for commonly used commands, especially those involving a '-' (which is virtually all commands that need input args/flags), since hitting the '-' requires a hand movement that gets my other fingers out of the base position "asdf-jkl;" (and when I bring them back, I am sometimes a bit off, and press the wrong keys, and have to look at the keyboard to adjust). E.g., ll='ls -l', llh='ls -lh, pmake='make -jN' (I set N depending on the machine I'm on, to leave 1-2 cores available for other stuff), fn='find . -follow -name'. And one of my favorites: mpigdb2='mpirun -np 2 xterm -e gdb --args', which you use to run gdb for mpi programs; I used to forget the syntax when I was first using it, so the alias helped.
- Git supports aliases too, so I have a variety of aliases in my .gitconfig (see attached). The ones I use the most are `git bkp` and `git ssu` , since they take a lot of keystrokes away, and I use them a lot when I need to switch back and forth between branches I'm working on (we have lots of submodules in e3sm, so there's always one to update).
- In my bashrc, I modified PS1 to show the branch name (if in a git repo):

  ```
  # Function that retrieves the current branch name if we're in a git repo tree
  function parse_git_branch {
    command git branch --no-color 2> /dev/null | sed -e '/^[^*]/d' -e 's/* \(.*\)/\ (\1)/'
  }

  PS1="${GREEN}\u@\h:[\W${BROWN}\$(parse_git_branch)${GREEN}]\$ "
  ```
- The .gdbinit script is also something that can make debugging a bit easier (see attached). In particular the line to obtain pretty print, and the line to print the actual (derived) object when printing the content of a pointer. A humanly-readable debugger output makes debugging orders of magnitude faster (I don't know why print pretty is not the default, honestly).
- I recently got into using lazygit. It's a terminal UI for git, easy to install (e.g, conda) and use. The feature I use the most is to commit individual lines in a file. You can of course do that with native git commands (after all, lg is running git in the background), but who wants to remember (and type!) the syntax? I also use it for solving merge conflicts, although some 3-panels tools (e.g., xxdiff, meld) are more powerful. There are other

nice similar tools (tig, gitui), but I find lazygit mature and robust, as well as very well maintained (minor releases come out somewhat often on conda-forge).

- I use vim for text editor, and I use some interesting plugins, which I recommend (I'm sure emacs has equivalent plugins):
    - yankring: allows to copy in one vim instance, and paste in others
    - YouCompleteMe: a code-completion that compiles code in the background, so you spot typos/errors while writing
    - nerdcommenter: allows to comment/uncomment line(s) of code, in several languages (bash, C/C++/Fortran, cmake, ...)