zsh-history-substring-search

This is a clean-room implementation of the Fish shell's history search feature, where you can type in any part of any command from history and then press chosen keys, such as the UP and DOWN arrows, to cycle through matches.

Requirements

- ZSH 4.3 or newer

Install

Using the Homebrew package manager:

```
brew install zsh-history-substring-search
echo 'source /usr/local/share/zsh-history-substring-search/zsh-history-substring-search.zsh' >> ~/.zshrc
```

Using Oh-my-zsh:

1. Clone this repository in oh-my-zsh's plugins directory:

   ```
   git clone https://github.com/zsh-users/zsh-history-substring-search ${ZSH_CUSTOM:-~/.oh-my-zsh/custom}/plugins/zsh-history-substring-search
   ```

2. Activate the plugin in `~/.zshrc`:

   ```
   plugins=( [plugins...] history-substring-search)
   ```

3. Source `~/.zshrc` to take changes into account:

   ```
   source ~/.zshrc
   ```

Usage

1. Load this script into your interactive ZSH session:

   ```
   % source zsh-history-substring-search.zsh
   ```

   If you want to use zsh-syntax-highlighting along with this script, then make sure that you load it *before* you load this script:

   ```
   % source zsh-syntax-highlighting.zsh
   % source zsh-history-substring-search.zsh
   ```

2. Bind keyboard shortcuts to this script's functions.

   Users typically bind their UP and DOWN arrow keys to this script, thus:

   - Run `cat -v` in your favorite terminal emulator to observe key codes.

(**NOTE:** In some cases, `cat -v` shows the wrong key codes. If the key codes shown by `cat -v` don't work for you, press `<C-v><UP>` and `<C-v><DOWN>` at your ZSH command line prompt for correct key codes.) * Press the

UP arrow key and observe what is printed in your terminal. * Press the DOWN arrow key and observe what is printed in your terminal. * Press the Control and C keys simultaneously to terminate the `cat -v`. * Use your observations from the previous steps to create key bindings. For example, if you observed `^[[A` for UP and `^[[B` for DOWN, then:

```
    bindkey '^[[A' history-substring-search-up
    bindkey '^[[B' history-substring-search-down

However, if the observed values don't work, you can try using terminfo:

    bindkey "$terminfo[kcuu1]" history-substring-search-up
    bindkey "$terminfo[kcud1]" history-substring-search-down

You might also want to bind the Control-P/N keys for use in EMACS mode:

    bindkey -M emacs '^P' history-substring-search-up
    bindkey -M emacs '^N' history-substring-search-down

You might also want to bind the `k` and `j` keys for use in VI mode:

    bindkey -M vicmd 'k' history-substring-search-up
    bindkey -M vicmd 'j' history-substring-search-down
```

3. Type any part of any previous command and then:
    - Press the `history-substring-search-up` key, which was configured in step 2 above, to select the nearest command that (1) contains your query and (2) is also older than the current command in your command history.
    - Press the `history-substring-search-down` key, which was configured in step 2 above, to select the nearest command that (1) contains your query and (2) is also newer than the current command in your command history.
    - Press `^U` the Control and U keys simultaneously to abort the search.
4. If a matching command spans more than one line of text, press the LEFT arrow key to move the cursor away from the end of the command, and then:
    - Press the `history-substring-search-up` key, which was configured in step 2 above, to move the cursor to the line above the cursored line. When the cursor reaches the first line of the command, pressing the `history-substring-search-up` key again will cause this script to perform another search.
    - Press the `history-substring-search-down` key, which was configured in step 2 above, to move the cursor to the line below the cursored line. When the cursor reaches the last line of the command, pressing the `history-substring-search-down` key, which was configured in step 2 above, again will cause this script to perform another search.

Configuration

This script defines the following global variables. You may override their default values only after having loaded this script into your ZSH session.

- `HISTORY_SUBSTRING_SEARCH_HIGHLIGHT_FOUND` is a global variable that defines how the query should be highlighted inside a matching command. Its default value causes this script to highlight using bold, white text on a magenta background. See the "Character Highlighting" section in the zshzle(1) man page to learn about the kinds of values you may assign to this variable.
- `HISTORY_SUBSTRING_SEARCH_HIGHLIGHT_NOT_FOUND` is a global variable that defines how the query should be highlighted when no commands in the history match it. Its default value causes this script to highlight using bold, white text on a red background. See the "Character Highlighting" section in the zshzle(1) man page to learn about the kinds of values you may assign to this variable.
- `HISTORY_SUBSTRING_SEARCH_GLOBBING_FLAGS` is a global variable that defines how the command history will be searched for your query. Its default value causes this script to perform a case-insensitive search. See the "Globbing Flags" section in the zshexpn(1) man page to learn about the kinds of values you may assign to this variable.
- `HISTORY_SUBSTRING_SEARCH_FUZZY` is a global variable that defines how the command history will be searched for your query. If set to a non-empty value, causes this script to perform a fuzzy search by words, matching in given order e.g. `ab c` will match `*ab*c*`
- `HISTORY_SUBSTRING_SEARCH_ENSURE_UNIQUE` is a global variable that defines whether all search results returned are *unique*. If set to a non-empty value, then only unique search results are presented. This behaviour is off by default. An alternative way to ensure that search results are unique is to use `setopt HIST_IGNORE_ALL_DUPS`. If this configuration variable is off and `setopt HIST_IGNORE_ALL_DUPS` is unset, then `setopt HIST_FIND_NO_DUPS` is still respected and it makes this script skip duplicate *adjacent* search results as you cycle through them, but this does not guarantee that search results are unique: if your search results were "Dog", "Dog", "HotDog", "Dog", then cycling them gives "Dog", "HotDog", "Dog". Notice that the "Dog" search result appeared twice as you cycled through them. If you wish to receive globally unique search results only once, then use this configuration variable, or use `setopt HIST_IGNORE_ALL_DUPS`.

History

- September 2009: Peter Stephenson originally wrote this script and it published to the zsh-users mailing list.
- January 2011: Guido van Steen (@guidovansteen) revised this script and released it under the 3-clause BSD license as part of fizsh, the Friendly Interactive ZSHell.
- January 2011: Suraj N. Kurapati (@sunaku) extracted this script from fizsh 1.0.1, refactored it heavily, and finally repackaged it as an oh-my-zsh plugin and as an independently loadable ZSH script.
- July 2011: Guido van Steen, Suraj N. Kurapati, and Sorin Ionescu (@sorin-ionescu) further developed it with Vincent Guerci (@vguerci).
- March 2016: Geza Lore (@gezalore) greatly refactored it in pull request #55.

Oh My Zsh Distribution Notes

What you are looking at now is Oh My Zsh's repackaging of zsh-history-substring-search as an OMZ module inside the Oh My Zsh distribution.

The upstream repo, zsh-users/zsh-history-substring-search, can be found on GitHub at https://github.com/zsh-users/zsh-history-substring-search.

This downstream copy was last updated from the following upstream commit:

SHA: 0f80b8eb3368b46e5e573c1d91ae69eb095db3fb Commit date: 2019-05-12 17:35:54 -0700

Everything above this section is a copy of the original upstream's README, so things may differ slightly when you're using this inside OMZ. In particular, you do not need to set up key bindings for the up and down arrows yourself in `~/.zshrc`; the OMZ plugin does that for you. You may still want to set up additional emacs- or vi-specific bindings as mentioned above.