



Mastering Quickfix

VimConf 2024
daisuzu

About me

- daisuzu(Daisuke Suzuki)
 -  https://x.com/dice_zu
 -  <https://github.com/daisuzu>
 -  <https://daisuzu.hatenablog.com>
- Job
 - Server side software engineer
 - Experienced in large-scale refactorings exceeding +/- 10,000 lines of code changes using Vim
- VimConf
 - 2017: How ordinary Vim user contributed to Vim
 - 2018: Migrating plugins to standard features
 - 2019: Usage and manipulation of the tag stack
- gorilla.vim
 - Frequently found at the reception desk

Introduction

Quickfix, in one sentence, is a list of jump targets.

- Often overlooked or considered legacy due to its non-interactive nature
- Actually a powerful and versatile feature
 - Useful for task management
 - Invaluable for large-scale refactoring

Vim's True Strength:

✗ Writing new code (LLMs excel here nowadays)

✓ Editing efficiency (Edit at the speed of thought)



The thrill of lightning-fast editing makes Vim addictive !

Agenda

1. Quickfix Basics
2. Advanced Techniques

Quickfix Basics

Basic Operations

List Creation:

- `:make` - Compile and capture errors
- `:grep` - Search files
- `:vimgrep` - Vim's internal grep
- `:helpgrep` - Search Vim help files

Managing quickfix window:

- `:copen` - Open the quickfix window
- `:cclose` - Close the quickfix window

Jumping:

- `:cc` - Jump to entry
 - `:cc [nr]` - Jump to specific entry
- `<CR>` - Jump to entry under the cursor
 - `CTRL-W <CR>` - Jump with new window
- `:cnext` / `:cprevious` - Jump to next/previous entry

Additional quickfix commands

Managing quickfix window:

- `:cwindow` - Open the quickfix window if there are entries, and close if none

Jumping:

- `:cbelow` / `:cabove` - Jump to the entry below/above the current line
- `:cafter` / `:cbefore` - Jump to the entry after/before the current position(line/column)
- `:cnfile` / `:cpfile` - Jump to the entry in next/previous file
- `:cfirst` (`:crewind`) / `:clast` - Jump to first/last entry

Quickfix history management

- `:chistory` - Display quickfix list stack
- `:colder` - Go to older quickfix list
- `:cnewer` - Go to newer quickfix list

Customizing quickfix

- `:set makeprg` - Customize the make program
 - `:set makeprg=staticcheck`
- `:set grepprg` - Customize the grep program
 - `:set grepprg=git\ grep\ -n\ --no-color`
- `:set errorformat` - Specifies a list of formats to parse
 - `:set errorformat=%f\|%l\ col\ %c-%k\|\ %m`
 - %f - file name
 - %l - line number
 - %c - column number
 - %k - end column number
 - %m - error message

Batch operations on quickfix

- `:cdo` - Execute commands for each entry
 - `:cdo s/OLD/NEW/g | w`
- `:cfdo` - Execute commands for each file
 - `:cfdo %s/OLD/NEW/g | w`

Useful plugins

Cfilter

- Bundled plugin to reduce the number of entries
 - `:packadd cfilter`
 - `:Cfilter /{pat}/` or `:Cfilter! /{pat}/`

qfreplace

- To perform the replacement in quickfix
 - `:Qfreplace`

Location list

Almost the same as quickfix, but differs in a few key aspects:

- Quickfix:
 - Global list for errors or search results across multiple files
 - Commands use 'c' prefix (e.g., `:copen`, `:cnext`, `:Cfilter`)
- Location list:
 - Local list for errors or search results within each individual window
 - Multiple location lists can be open simultaneously in different windows
 - Commands use 'l' prefix (e.g., `:lopen`, `:lnext`, `:Lfilter`)

| | |
|-----------|-----------|
| Buffer #1 | Buffer #2 |
| Quickfix | |

| | |
|------------------|------------------|
| Buffer #1 | Buffer #2 |
| Location list #1 | Location list #2 |

Advanced Techniques

Saving and loading quickfix lists

Saving:

- Write quickfix buffer
 - `:w filename`

Loading:

- `:cfile` / `:cgetfile` - Read from file
 - `:cfile filename` / `:cgetfile filename`
- `:cbuffer` / `:cgetbuffer` - Read from buffer
 - `:e filename`
 - `:cbuffer` / `:cgetbuffer`

Automating with macros

1. `qq` - Start recording
2. Perform your operations
3. `:w` - Write changes
4. `:cnext` - Move to next entry
5. `q` - Stop recording
6. `@q / 10@q` - Execute macro

Real-World Example:

Scenario: Expand i18n templates using multiple language dictionaries

Before:

```
{{ i18n "hello" }} {{ .name }}  
{{ i18n "goodbye" }} {{ .name }}
```

Dictionaries:

| File | Content |
|-----------|-----------------|
| intro.en | hello=Hello |
| intro.ja | hello=こんにちは |
| ending.en | goodbye=Goodbye |
| ending.ja | goodbye=さようなら |

After:

```
こんにちは {{ .name }}  
さようなら {{ .name }}
```

Dictionaries:

| File | Content |
|------|---------|
| | |

Operation

Preparation:

```
function! Expand_i18n()  
  " Yank target keyword  
  normal f"  
  normal "ayi"  
  
  " Extract translation message from dictionary files  
  execute "lgrep! '^" .. @a .. "=" -- *.ja"  
  let l:loclist = getloclist('.')  
  if empty(l:loclist)  
    return  
  endif  
  let @b = l:loclist  
    \ ->map('split(v:val.text, "=")')  
    \ ->filter('len(v:val) == 2')  
    \ ->map('v:val[1]')[0]  
  
  " Expand template  
  normal 3B  
  execute 'normal v%"bp'  
endfunction
```

Execution:

```
" Find target lines  
:grep '{{{ i18n'  
  
" Record macro  
qq  
:call Expand_i18n()  
:w  
:cnext  
q  
  
" Execute macro  
1000@q
```

The Essence of Vim Mastery

Mastering Vim means thinking in Vim commands and editing at the speed of thought.

To achieve this, focus on these key points:

1. Break down actions into “motion” and “object”
2. Expand your vocabulary for moving around
3. Turn complex actions into simple commands
4. Be aware of repeatable actions

By practicing these principles continuously, you can truly master Vim.

Summary

- Quickfix is a powerful feature for managing lists of locations in your code
 - It excels in non-interactive, reproducible workflows
 - Plugins like Cfilter and qfreplace enhance quickfix functionality
- Combining quickfix with macros enables complex, automated text processing
 - Mastering quickfix can significantly boost your productivity in Vim
- Striving for efficient editing leads to mastering Vim