Usage and manipulation of the tag stack
About me

● daisuzu(Daisuke Suzuki)
  ○ ✨ https://twitter.com/dice_zu
  ○ 🗠 https://github.com/daisuzu
  ○ ✍️ https://daisuzu.hatenablog.com

● Job
  ○ Server side software engineer
  ○ Go Language Committee(in-company organization)

● VimConf

How ordinary Vim user contributed to Vim

VimConf 2017
daisuzu

2017

Migrating plugins to standard features

VimConf 2018
daisuzu
twitter: @dice_zu

2018
Motivation

- I had to submit for CFP, but there was no content to talk.
- By the way, I love vim-lsp and wanted to use built-in tag jump with vim-lsp.
  - prabirshrestha/vim-lsp#435 by @kailin4u (Merged on July 18)
    - But there is a problem that go back too much
- I investigated the internal processing of the tag stack and fixed it.
  - prabirshrestha/vim-lsp#449 by @daisuzu (Created on July 26)
  - Tag stack is really deep, so I decided to talk about this

Presentation Driven Development
Agenda

● Basics of the tags feature
● Deep dive into the tag stack
Basics of the tags feature
What is a tag?

- It is a location where an identifier is defined
  - That’s news to you? Please see :help

```
help.txt     For Vim version 8.1. Last change: 2019 Jul 21

VIM- main help file

Move around: Use the cursor keys, or "h" to go left, "l" to go right.
            "j" to go down, "k" to go up.
Close this window: Use ":q<Enter>".
Get out of Vim: Use ":qa!<Enter>" (careful, all changes are lost!).

Jump to a subject: Position the cursor on a tag (e.g. bars) and hit CTRL-].
```

- A list of tags is kept in a tags file
  - The tags file has to be generated before the tag commands can be used
    - ctags
    - :helptags
Using tags

- **Preparations**
  1. Generate the tags file
     ```
     # Supported languages like C/C++
     $ ctags -R
     ```
  2. Set tags option (if need to change)
     ```
     :set tags=./tags,tags
     ```

- **Jump to tag**
  `CTRL-]`

- **Go back**
  `CTRL-T`
Example) Jump to tag

```c
void write_block(char **s; int cnt) {
    int i;
    for (i = 0; i < cnt; ++i)
        write_line(s[i]);
}

void write_line(char *s) {
    while (*s != 0)
        write_char(*s++);
}
```
Example) Jump to tag

```c
void write_block(char **s; int cnt)
{
    int i;
    for (i = 0; i < cnt; ++i)
        write_line(s[i]);
}

void write_line(char *s)
{
    while (*s != 0)
        write_char(*s++);
}

void write_char(char c)
{
    putchar((int)(unsigned char)c);
}
```
Example) Show the contents of the tag stack

```c
void write_block(char **s; int cnt)
{
    int i;
    for (i = 0; i < cnt; ++i)
        write_line(s[i]);
}

void write_line(char *s)
{
    while (*s != 0)
        write_char(*s);
}

void write_char(char c)
{
    putchar((int)(unsigned char)c);
}
```

```
# TO tag FROM line in file/text
1 1 write_line 8 write_block.c
2 1 write_char 7 write_line.c
```
```c
void write_block(char **s; int cnt)
{
    int i;
    for (i = 0; i < cnt; ++i)
        write_line(s[i]);
}

void write_line(char *s)
{
    while (*s != 0)
        write_char(*s++);
}

void write_char(char c)
{
    putchar((int)(unsigned char)c);
}
```
```c
void write_block(char **s; int cnt)
{
    int i;
    for (i = 0; i < cnt; ++i)
        write_line(s[i]);
}

void write_line(char *s)
{
    while (*s != 0)
        write_char(*s++);
}

void write_char(char c)
{
    putchar((int)(unsigned char)c);
}
```
## Tag commands

### Jump to tag

- **in same window**
  - CTRL-], {Visual}CTRL-
  - 
    | [count] | tag | [name] |
    |--------|-----|--------|
  - g<LeftMouse>, <C-LeftMouse>
- **in split window**
  - :stag [name]
  - CTRL-W CTRL-], CTRL-W ]
- **in preview window**
  - :ptag [name]
  - CTRL-W ]
- and add the matching tags to a new location list
  - :ltag [name]

### Go back

- **CTRL-T**
- 
<table>
<thead>
<tr>
<th>[count]</th>
<th>pop</th>
</tr>
</thead>
</table>
  - :[count]pop
  - g<RightMouse>, <C-RightMouse>
- 
<table>
<thead>
<tr>
<th>[count]</th>
<th>ppop</th>
</tr>
</thead>
</table>
  - :[count]ppop

---

Go back
Tag match list

Select a tag to jump

● in same window
  ○ :tselect [name]
  ○ g] , {Visual}g]

● in split window
  ○ :stselect [name]
  ○ CTRL-W g ]

● in preview window
  ○ :ptselect [name]

Jump directly when there is only one match, otherwise select a tag to jump

○ :tjump [name]
○ g CTRL-] , {Visual}g CTRL-

○ :stjump [name]
○ CTRL-W g CTRL-

○ :ptjump [name]
○ CTRL-W g }
Difference from the jump-motions

The jump-motions jumps around the jump list with **CTRL-I** or **CTRL-O**.

- **CTRL-I** ... Go to newer cursor position in jump list
  - Cannot move to a new location alone
  - In other words, the location must be changed using other commands
    - tag jump
    - search
    - goto line

- **CTRL-O** ... Go to older cursor position in jump list
  - It will go back including other commands as above
Deep dive into the tag stack
Tag stack structure

Store the following data for each window:

<table>
<thead>
<tr>
<th>items</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>items</td>
<td>List of items in the stack.</td>
</tr>
<tr>
<td>tagname</td>
<td>Name of the tag.</td>
</tr>
<tr>
<td>from</td>
<td>Cursor position before the tag jump.</td>
</tr>
<tr>
<td>matchnr</td>
<td>Current matching tag number.</td>
</tr>
<tr>
<td>bufnr</td>
<td>Buffer number of the current jump.</td>
</tr>
<tr>
<td>user_data</td>
<td>Custom data string for 'tagfunc'.</td>
</tr>
<tr>
<td>curidx</td>
<td>Current index in the stack. Index of bottom of the stack is 1.</td>
</tr>
<tr>
<td>length</td>
<td>Number of entries in the stack.</td>
</tr>
</tbody>
</table>
Jump

length = 0

curidx = 1

CTRL-]

curidx = 2

length = 1

<table>
<thead>
<tr>
<th>tagname</th>
<th>'vim_main2'</th>
</tr>
</thead>
<tbody>
<tr>
<td>from</td>
<td>[1, 444, 12, 0]</td>
</tr>
<tr>
<td>matchnr</td>
<td>1</td>
</tr>
<tr>
<td>bufnr</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>tagname</td>
</tr>
<tr>
<td>---</td>
<td>---------</td>
</tr>
<tr>
<td>1</td>
<td>from</td>
</tr>
<tr>
<td></td>
<td>matchnr</td>
</tr>
<tr>
<td></td>
<td>bufnr</td>
</tr>
</tbody>
</table>

Ctrl-T

<table>
<thead>
<tr>
<th></th>
<th>tagname</th>
<th>'vim_main2'</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>from</td>
<td>[1, 444, 12, 0]</td>
</tr>
<tr>
<td></td>
<td>matchnr</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>bufnr</td>
<td>1</td>
</tr>
</tbody>
</table>

curidx = 2

curidx = 1
Jump again

length = 1

tagname: 'vim_main2'
from: [1, 442, 12, 0]
matchnr: 1
bufnr: 1
curidx = 1

CTRL-[

tagname: 'mzscheme_main'
from: [1, 442, 12, 0]
matchnr: 1
bufnr: 1

curidx = 2
Replace top \((\text{length} == \text{curidx})\)

- **length = 2**
  - curidx = 3
  - BBB
  - AAA

- **length = 2**
  - curidx = 2
  - BBB
  - AAA

- **length = 2**
  - curidx = 3
  - CCC
  - AAA

CTRL-T

CTRL-]
Replace all \((\text{length} > \text{curidx})\)
Remove and replace ($length > curidx$)

- $curidx = 4$
- $curidx = 3$
- $curidx = 2$
- $curidx = 3$
How the tag stack is managed

- **When jump**
  - Add or replace *items*
    - *items* above *curidx* are removed
    - *length* will also be changed
  - *curidx*++

- **When go back**
  - *curidx*--
Manipulate the tag stack from vim script

`settagstack()` added in v8.1.0519 made it possible.

- Works without the tags file
- That means it can be used with LSP

```
settagstack({nr}, {dict} [, {action}])
```

- **nr** ... The window number or the window-ID of the target
- **dict** ... A dictionary of the tag stack structure except `length`
- **action**
  - **r** ... Replace items in the current tag stack with `dict.items` (default)
  - **a** ... Append `dict.items` to the current tag stack

Paired function is `gettagstack()`.

But this is a very low-level function, so `items` and `curidx` must be handled correctly.
Implementation example

```plaintext
1 let bufnr = bufnr('%')
2 let item = {
3   'bufnr': bufnr,
4   'from': [bufnr, line('.'), col('.')],
5   'tagname': expand('<cword>')
6 }
7 let nr = win_getid()

" Update tag stack to be set.
+-- 16 lines: let dict = gettagstack(nr)-----------------------------

26 call settagstack(nr, dict, action)
```
Implementation example

```
10 let dict = gettagstack(nr)
11 if dict['length'] == dict['curidx']
12   let action = 'r'
13   let dict['items'][dict['curidx']-1] = item
14 elseif dict['length'] > dict['curidx']
15   let action = 'r'
16   if dict['curidx'] > 1
17     let dict['items'] = add(dict['items'][:dict['curidx']-2], item)
18   else
19     let dict['items'] = [item]
20   endif
21 else
22   let action = 'a'
23   let dict['items'] = [item]
24 endif
25 let dict['curidx'] += 1
```
Implementation example

```javascript
10 let dict = gettagstack(nr)
11 if dict['length'] == dict['curidx']   " Replace top
12  let action = 'r'
13  let dict['items'][dict['curidx']-1] = item
14  elseif dict['length'] > dict['curidx']
15    let action = 'r'
16    if dict['curidx'] > 1
17      let dict['items'] = add(dict['items'][:dict['curidx']-2], item)
18    else
19      let dict['items'] = [item]
20    endif
21  else
22    let action = 'a'
23    let dict['items'] = [item]
24  endif
25  let dict['curidx'] += 1
```
Implementation example

```javascript
let dict = gettagstack(nr)
if dict['length'] == dict['curidx']
  let action = 'r'
  let dict['items'][dict['curidx'] - 1] = item
elseif dict['length'] > dict['curidx']
  let action = 'r'
  if dict['curidx'] > 1
    let dict['items'] = add(dict['items'][:dict['curidx']-2], item)
  else
    let dict['items'] = [item] /* Replace all */
  endif
else
  let action = 'a'
  let dict['items'] = [item]
endif
let dict['curidx'] += 1
```

Implementation example

```python
10 let dict = gettagstack(nr)
11 if dict['length'] == dict['curidx']
12   let action = 'r'
13   let dict['items'][dict['curidx']-1] = item
14 elseif dict['length'] > dict['curidx']
15   let action = 'r'
16   if dict['curidx'] > 1
17     let dict['items'] = add(dict['items'][:dict['curidx']-2], item)
18   else
19     let dict['items'] = [item]
20 endif
21 else
22   let action = 'a'  " Append only"
23   let dict['items'] = [item]
24 endif
25 let dict['curidx'] += 1
```
Summary

- Tag stack is a very useful Vim basic feature for jumping between files
  - `CTRL-]` ... Jump to tag
  - `CTRL-T` ... Go back

- Can be used with LSP
  - By plugin that using `settagstack()`
  - Let’s map `CTRL-]`

- You can customize the tag stack behavior as you like, not just LSP