fly.vim

User's Guide

Revision 1.0

fly.vim@live.com

Table of Contents

1 What is *fly.vim*?

Are you using vim as your editor? If yes, *fly.vim* may be useful for you. fly.vim is a vim plugin that may enhance your vim experience (and productivity) with an integrated code development environment. The plugin facilitates rapid code-***walk****through* (or ***fly****through* as I call it), enables instant symbol searches and cross references in a large code base, manages stacks of multiple files and integrates code reviews and compilation, all within a simple and intuitive user interface.

fly.vim integrates seamlessly with source code index databases (Cscope for now), web search engines (Google and Yahoo) and UNIX man pages. Most functions are mapped to shortcut keys, allowing you to fly through a large code base or web in a convenient manner. Using the shortcut mappings, you may search for symbols (or multi-word phrases) after visually highlighting them, or by typing them on vim command line or by taking the cursor to the symbol in a vim window.

Although there are quite a few features to talk about later, it is very easy to get started,

- Download and place the fly.vim in your $HOME/.vim/plugin
- Change directory (cd) to the root of the source tree (or subtree)
- Launch vim. Start using the plugin.

2 What platforms?

The plugin has been in use on Linux and FreeBSD with vim 6.x and 7.y.

You are free to try it out on other platforms. Please make sure the dependencies are met. If you get it to work on other platforms, please let the plugin developer[s] know. (fly.vim@live.com)

3 What dependencies?

Dependencies are very few and in all likelihood your system probably has them already.

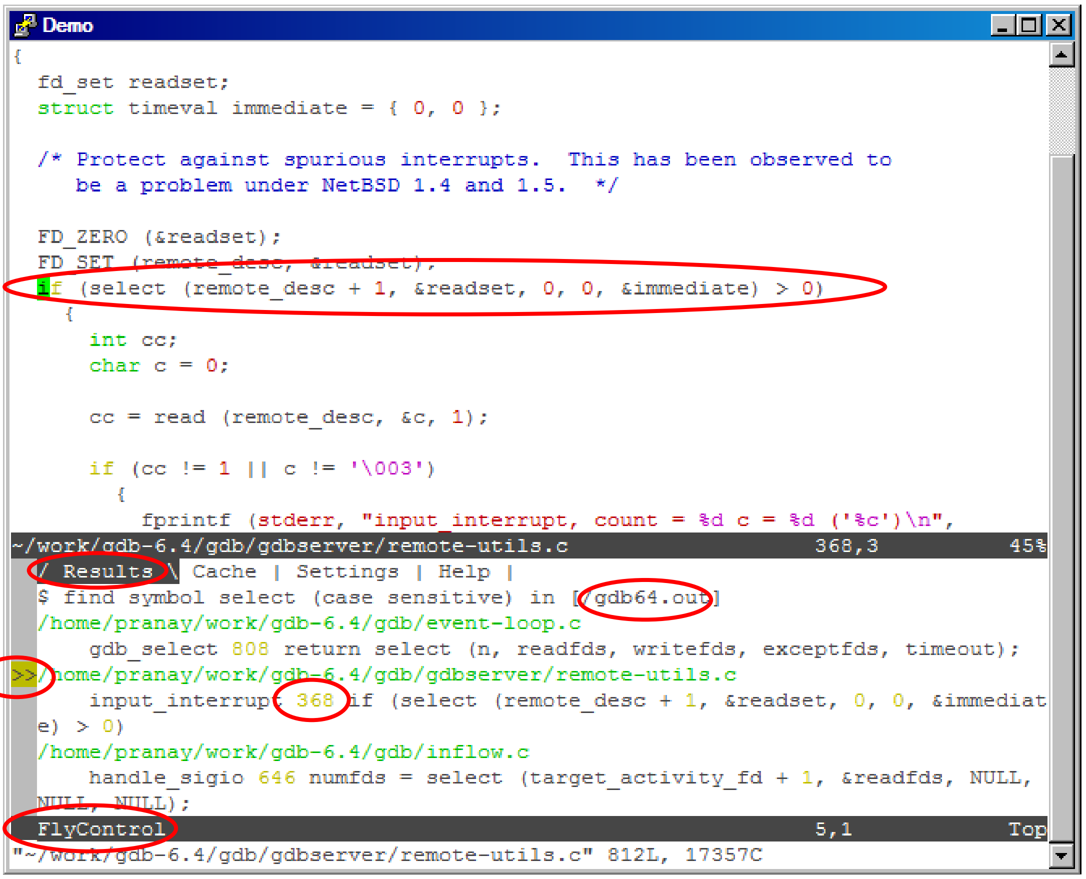
The plugin internally uses basic shell commands: *find, xargs, grep, cut, mkdir, man* that should be available on all UNIX shells. The plugin also uses open source tools *cscope* and *elinks*. Please make sure these are installed already.

4 How to install and use?

As usual, the plugin is installed in $HOME/.vim/plugin. After the installation please follow the steps in chapter 7 to get started. You may use chapter 6 as a Quick Reference Sheet for features.

5 Preview Please?

The following page shows a sample screenshot. More screenshots can be found later in this guide where individual features are described. The plugin has a control window to access most features and also a pre-defined set of mappings to run most commands. Results for most commands show up on the plugin's control window – referred to as *fly control window* hereafter. Display for the fly control window could be toggled anytime.



```
{
  fd_set readset;
  struct timeval immediate = { 0, 0 };

  /* Protect against spurious interrupts.  This has been observed to
     be a problem under NetBSD 1.4 and 1.5.  */

  FD_ZERO (&readset);
  FD_SET (remote_desc, &readset);
  if (select (remote_desc + 1, &readset, 0, 0, &immediate) > 0)
    {
      int cc;
      char c = 0;

      cc = read (remote_desc, &c, 1);

      if (cc != 1 || c != '\003')
        {
          fprintf (stderr, "input_interrupt, count = %d c = %d ('%c')\n",
~/work/gdb-6.4/gdb/gdbserver/remote-utils.c                   368,3           45%
  / Results \ Cache | Settings | Help |
  $ find symbol select (case sensitive) in [/gdb64.out]
  /home/pranay/work/gdb-6.4/gdb/event-loop.c
      gdb_select 808 return select (n, readfds, writefds, exceptfds, timeout);
>>/home/pranay/work/gdb-6.4/gdb/gdbserver/remote-utils.c
      input_interrupt 368 if (select (remote_desc + 1, &readset, 0, 0, &immediat
  e) > 0)
  /home/pranay/work/gdb-6.4/gdb/inflow.c
      handle_sigio 646 numfds = select (target_activity_fd + 1, &readfds, NULL,
  NULL, NULL);
  FlyControl                                                  5,1             Top
"~/work/gdb-6.4/gdb/gdbserver/remote-utils.c" 812L, 17357C
```

For all commands, user may select any one of the results just by hitting an <Enter> key on the result line in fly control window. The intent is to provide a unified UI for most features. Most often a result line is associated with a tuple <filename, line number>. A hit on <Enter> key on result line opens associated file for editing and takes you to that line number. For web searches the result line corresponds to a URL and a hit on that line opens up the URL in a separate vim window. The sample screenshot shows results for a Cscope symbol search command in the fly control window and also shows that the file associated with selected result line is opened in an edit window.

6 Features Quick Reference?

fly command set is summarized in the following table. These commands use upper case letters to minimize chances of conflicts with native vim commands or other plugin commands. In case you prefer to use lower case letters for commands, you may chose to redefine them. Most commands also have pre-defined mappings (short cut keys).

It is recommended to use vim mapping (when defined) for a feature. Almost all mappings use the vim <*Leader*> character. Default vim <*Leader*> *is* **'\'.** You may define your own mappings if there is a conflict with existing set of mappings in your vim environment or otherwise also as per personal taste. The mappings are defined at the tail end of plugin source file, so that it's easy to locate and change them when you want to do so.

You may also redefine the <Leader> as per personal preference e.g.

```
:let mapleader = ","
```

Command	**Mapping**	**Description**
SW	<Leader>y	Toggle the display of fly control window.
		Cscope commands
LD		**L**ist Cscope **d**atabases pre-built in $CSCOPE_DIR.
BD [name]		**B**uild Cscope **d**atabase for the source tree rooted in current directory. The <name> for the database is optional, default name is last part of directory name with extension ".out". You may build the database for a specific source root specified by `g:root="…"`.
CS		Query in currently selected **CS**cope Database. The mapping chars [sgctefid] are intentionally kept same as those in *cscope.sourceforge.net/cscope_maps.vim*
The same mappings in visual mode apply to visually highlighted text.		
CS s <sym>	<Leader><Leader>s	Search for occurrences of symbol *at cursor*
CS g <sym>	<Leader><Leader>g	Search for definition of the symbol *at cursor*
CS c <sym>	<Leader><Leader>c	Search for calls to the function *at cursor*
CS t <sym>	<Leader><Leader>t	Search for text *at cursor*
CS e <sym>	<Leader><Leader>e	Egrep Search for the symbol *at cursor*
CS f <sym>	<Leader><Leader>f	Search for file *at cursor*
CS i <sym>	<Leader><Leader>i	Search for files that include the files *at cursor*
CS d <sym>	<Leader><Leader>d	Search for functions called by function *at cursor*
CS s <sym>	<Leader>s	Search for occurrences of <sym> *typed by user*
CS g <sym>	<Leader>g	Search for definition of the <sym> *typed by user*
CS c <sym>	<Leader>c	Search for calls to the function <sym> *typed by user*
CS t <sym>	<Leader>t	Search for text <sym> *typed by user*
CS e <sym>	<Leader>e	Egrep Search for the <sym> *typed by user*
CS f <sym>	<Leader>f	Search for file <sym> *typed by user*
CS i <sym>	<Leader>i	Search for files including file <sym> *typed by user*
CS d <sym>	<Leader>d	Search for functions called by <sym> *typed by user*
		List Commands
LS	<Leader>ls	**List** all files in directory containing the opened file in current edit window in focus.
LS <path>		**List** all files in the directory <path>, where <path> is

		absolute path.
LB	<Leader>lb	**L**ist all **b**uffers currently open in vim
		Grep Commands
GR [opt] <sym>		**Gr**ep for <sym> in all the files currently listed in the fly control window. The options are command line options to UNIX *grep*. The command currently works only for results of CS/LS/GR command.
		Build Commands
MK	<Leader>b	Build/compile source. You must have done "cd" at vim command line to the dir where builds are done. Also, makeprg should have been set if applicable. E.g. :set makeprg=gmake
		Read File Command
RD <file>		• Read <file> containing a list of files. • Read <file> containing log of a build/compilation.
		Web Commands
WS	<Leader><Leader>w	**W**eb **S**earch for symbol under cursor or visually highlighted text. There are two choices for internet search engines as of now – Google and Yahoo. A choice for the search engine and number of search results can be made in Settings Tab Toggle section. Default search engine is Google. Default number of search results is 10.
WS <text>	<Leader>w	**W**eb **S**earch for <text> typed at the command line.
OW	<Leader><Leader>o	**O**pen a **W**eb URL that is present in current line.
OW <text>	<Leader>o	Open a URL that is typed on the command line. The URL regular expression used in the plugin works in most common cases, far from perfection though.
TW		Toggle the web page window.
		Man Pages
MAN [N] <sym>	<Leader>m	Get UNIX man page for <sym> in a vim window. Its useful to yank #include <file> and example code for system calls and library functions in the man pages.
	<Leader>m***N***	Argument N is optional. Get the UNIX man section ***N***. ***N*** **€** {1..8}
TM		Toggle Man Page Window.
		Stack Commands
		The stack commands apply to editing window, results tab, Man pages window and Web pages window.
LF	<Leader>**,**	Go down the buffer stack in current window in focus. (Imagine using a “<” **Left** arrow for going down the stack.)
RT	<Leader>**.**	Go up the buffer stack in current window in focus. (Imagine using a “>” **Right** arrow for going up the

		stack)
LK		**L**ist the contents of buffer stac**k** in current edit window in focus. (Currently) It is not supported for Web and Man pages window.
		Code Review/Sandbox Comparison Commands
TCVS		**T**oggle command to "**t**ie" or "un**t**ie" the current sandbox (source base selected by choice of Cscope database) to **CVS**. For every file that is opened in the edit window using fly commands, CVS vimdiff is shown as well in another window. The plugin *cvscommand.vim* must've been installed.
TSVN		**T**oggle command to "**t**ie" or "un**t**ie" the current sandbox (source base selected by choice of Cscope database) to **SVN**. For every file that is opened in the edit window using fly commands, SVN vimdiff is shown as well in another window. The plugin *svncommand.vim* must've been installed.
TIE		**T**oggle command to "**tie**" or "un**tie**" the current sandbox (source base selected by choice of Cscope database) to another sandbox specified by let g:sandbox="/absolute/path/to/root_of_sandbox/" For every file that is opened in the edit window using fly commands, vimdiff with corresponding file in the other sandbox is shown as well in another window.
DF	<Leader>z	Do a vim diff of file in current window of focus and corresponding file in the other window. This is useful when you do not want to TIE and do diff on demand.
:*diffget*	<Leader>l	Pull the diff from the other window to current window in focus.
:*diffput*	<Leader>p	Put the diff from the current window in focus to the other window.
		Session Save/Restore (Works only for results of CS commands for now)
:w <file>		Save contents of fly control window in a <file>.
RS <file>		**Res**tore the contents of <file> (that was saved above) in fly control window. And then you may jump to the result lines.
		Tab Navigation
	<Leader>1	Go to Results Tab
	<Leader>2	Go to Cache Tab
	<Leader>3	Go to Settings Tab

	<Leader>4	Go to Help Tab
		Commands in Settings Tab
		Toggle the case sensitivity of Cscope queries.
		Toggle the choice of Internet Search Engine (Google or Yahoo)
		Toggle the choice of internet search results (10, 15, 20, 30, 40, 100)

7 A few steps before you start,

7.1 Jump Start

I usually pre-build Cscope databases before starting vim, though this is not necessary. You may build database from the plugin itself using BD command. All you have to do is,

- Change directory to the root of the source tree (or subtree) that you want to work with.
- Launch vim. Run the BD command.

And you are all set to go. Cscope database is built in background and kept in $CSCOPE_DIR if it is set. Otherwise it is kept in the same directory as the root of the source.

7.2 Pre Build Databases

You may chose to pre-build all your Cscope databases that you usually work with and keep them in a reference directory. The environment variable CSCOPE_DIR must be set to this directory. The file listing used to build should preferably have *absolute path names* for files although that is not required for most commands. Cscope databases must be named with extensions *'.out'* or *'.db'*. For faster search it is recommended to build indexed databases using –q option in Cscope. The steps are summarized as follows,

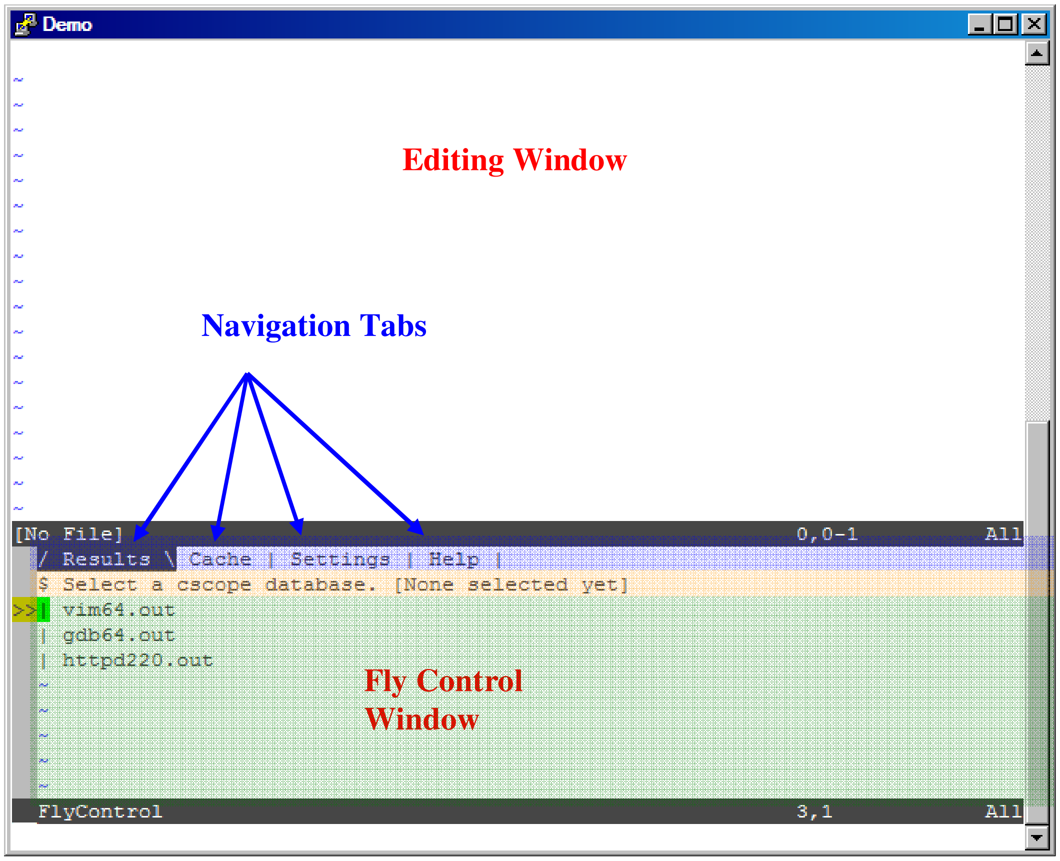
- Set the environment variable. `$ export CSCOPE_DIR=$HOME/cscope.dbs`
- Build Cscope databases.
  ```
  find /absolute/path/to/root_of_sandbox –name "pattern" > files.lst
  cscope –b –q –k –i files.lst –f $CSCOPE_DIR/dbname.db
  ```
- Launch vim and run LD command to list pre-built databases. Select a cscope database to use (just hit <enter> on a line listing the database that you want to work with)
- Start *fly*ing through the code!

8 After you start,

The fly.vim plugin functions are available as commands, pre-defined mappings and from tabs in the *fly control* window. The fly control window is *toggled* with the command '**:SW**' from vim command-line. The plugin has a default mapping of <Leader>y to do the same. After launching vim use this command to initialize the plugin. If you ran BD or LD as mentioned in last section, the initialization would have happened already.

The fly control window shows up at the bottom of all other windows opened in vim. The control window is a *read-only* vim window. Vim editing (*write*) commands in the control window are not expected by the plugin, however other vim commands (including movements and searches) should be okay.

Following screenshot shows the fly control window and default contents when CSCOPE_DIR is set in the host shell environment.



```
[No File]                                                 0,0-1          All
 / Results \ Cache | Settings | Help |
 $ Select a cscope database. [None selected yet]
>>| vim64.out
  | gdb64.out
  | httpd220.out
  ~
  ~
  ~
  ~
  ~
 FlyControl                                               3,1            All
```

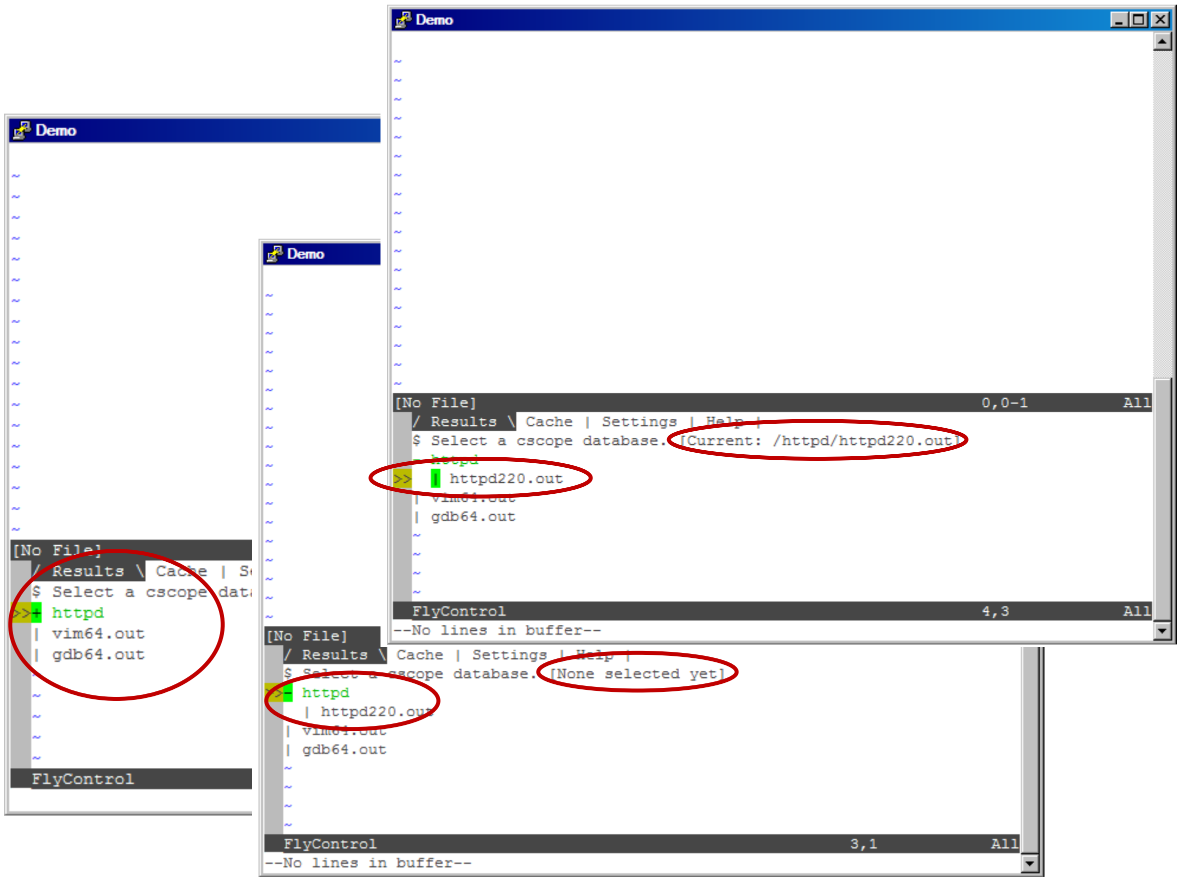
8.1 Navigation Tab Bar

The fly control window has a *Navigation Tab Bar* to select different choices offered by fly today. The current Tab[1] Selection is highlighted. Default Tab Selection is *Results.* Other Tabs could be selected by taking the cursor to the Tab text followed by <Enter>. The plug-in also has default mappings <Leader>1, <Leader>2, <Leader>3 and <Leader>4 to navigate through these Tabs.

8.1.1 Results Tab

This tab shows results of query commands. The top line in this Tab shows the command that was executed. Lines following that show the result lines that may be selected by just hitting <Enter>.

[1] The plugin was developed before the Tabs were introduced in vim and hence they may seem out of style here. Well, some day the fly developer[s] may enhance it to the use vim tabs.



```
Demo
[No File]
/ Results \ Cache | S
$ Select a cscope dat
>>+ httpd
  | vim64.out
  | gdb64.out
FlyControl
```

```
Demo
[No File]
/ Results \ Cache | Settings | Help |
$ Select a cscope database. [None selected yet]
>>- httpd
    | httpd220.out
  | vim64.out
  | gdb64.out
FlyControl                                        3,1          All
--No lines in buffer--
```

```
Demo
[No File]                                               0,0-1          All
/ Results \ Cache | Settings | Help |
$ Select a cscope database. [Current: /httpd/httpd220.out]
- httpd
>>  | httpd220.out
| vim64.out
| gdb64.out
FlyControl                                              4,3            All
--No lines in buffer--
```

Choosing a cscope database

The set of screenshot above demonstrate selection of a Cscope database. The plugin run **LD** command on the vim launch automatically. The LD command lists Cscope databases stored in CSCOPE_DIR. The databases could be organized in a sub-tree underneath CSCOPE_DIR as shown in this example. A flat file structure under CSCOPE_DIR is fine as well.

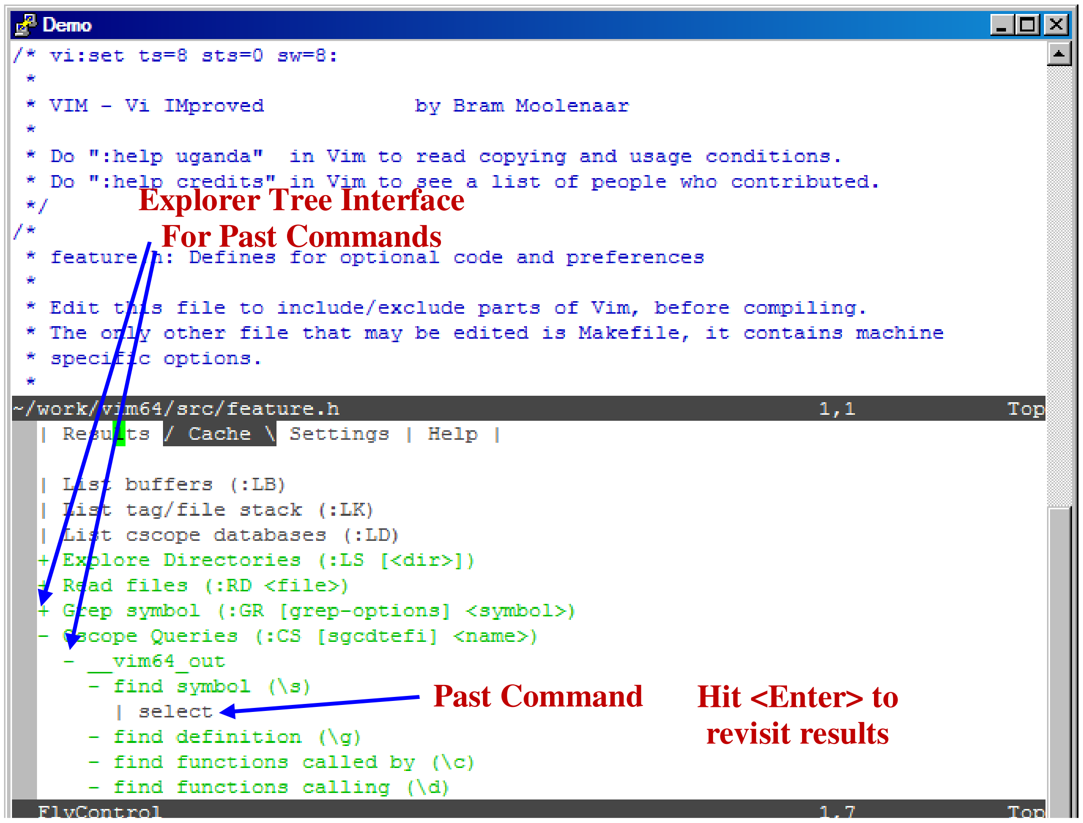
There is one Cscope database in this example that is kept in a sub-tree. The rest two are stored at CSCOPE_DIR level. The sub-trees are listed in traditional explorer way and could be expanded or collapsed just by <Enter>.

A Cscope database underneath a sub-tree could be selected by <Enter> and the selection would show up on the top line in fly control window. This is the Cscope database that will be used for subsequent Cscope queries. Cscope database selection could be changed at any point of time though.

For most fly commands the behavior is same as described in the example above. The command shows up on the topmost line in fly control window under Results tab and the results of the command show up on subsequent lines. Any of the result line could be selected (or activated) by <Enter> key.

8.1.2 Cache Tab

The cache tab displays the cache of the commands executed in the past. One may revisit the results of a previously executed command by just hitting an <Enter> on the line. For example in the following screenshot a search for a symbol 'select' was done in past on cscope database for vim64 sources. The search results could be revisited by <Enter> on the line showing the cache entry for file search 'Select'.



```
/* vi:set ts=8 sts=0 sw=8:
 *
 * VIM - Vi IMproved            by Bram Moolenaar
 *
 * Do ":help uganda"  in Vim to read copying and usage conditions.
 * Do ":help credits" in Vim to see a list of people who contributed.
 */
/*
 * feature.h: Defines for optional code and preferences
 *
 * Edit this file to include/exclude parts of Vim, before compiling.
 * The only other file that may be edited is Makefile, it contains machine
 * specific options.
 *
~/work/vim64/src/feature.h                                    1,1           Top
 | Results / Cache \ Settings | Help |

 | List buffers (:LB)
 | List tag/file stack (:LK)
 | List cscope databases (:LD)
 + Explore Directories (:LS [<dir>])
 + Read files (:RD <file>)
 + Grep symbol (:GR [grep-options] <symbol>)
 - Cscope Queries (:CS [sgcdtefi] <name>)
   - __vim64_out
     - find symbol (\s)
       | select
     - find definition (\g)
     - find functions called by (\c)
     - find functions calling (\d)
 FlyControl                                                   1,7           Top
```

Most recent command in the cache is highlighted with ">>". The cache tab also has windows explorer like interface to explore the cache entries.

For every Cscope database that was selected in past there is an explorer-sub-tree in the cache.

The cache is not supported for build commands and web commands for now. This cache is not persistent and is lost across vim launches.

8.1.3 Settings Tab

The Settings Tab shows current settings and environment for the plugin. It also allows one to change some of the plugin settings.

8.1.3.1 Environment

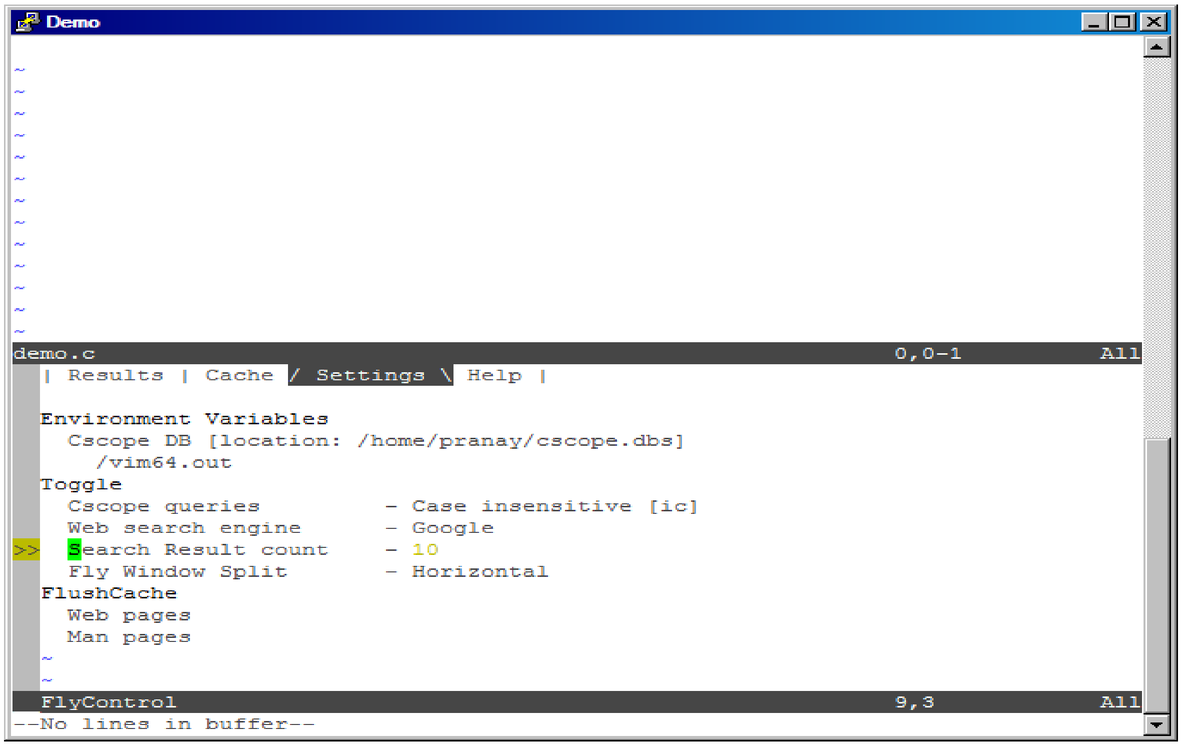
For now, there is only one environment variable CSCOPE_DIR that it shows in the environment section. It also shows currently selected Cscope database.

8.1.3.2 Toggle

In the toggle section, it allows one to toggle case-sensitivity of Cscope searches, toggle the choice of the Internet search engine (Google or Yahoo for now), toggle the count for Internet search results (10, 15,20, 30, 40, 100) and toggle the choice of split for fly windows (web/man). In order to toggle the choices, all one has to do is to hit <Enter> on that line and cycle through the choices.

8.1.3.3 Flush Persistent Cache

It also allows one to flush the cache of man pages and web pages. This cache is persistent as opposed to the command cache. In order to flush the cache, all one has to do is to hit <Enter> on that line.



```
~
~
~
~
~
~
~
~
~
~
~
~
~
demo.c                                                           0,0-1          All
  | Results | Cache / Settings \ Help |

  Environment Variables
    Cscope DB [location: /home/pranay/cscope.dbs]
      /vim64.out
  Toggle
    Cscope queries          - Case insensitive [ic]
    Web search engine       - Google
>>  Search Result count     - 10
    Fly Window Split        - Horizontal
  FlushCache
    Web pages
    Man pages
  ~
  ~
  FlyControl                                                     9,3            All
--No lines in buffer--
```

8.1.4 Help Tab

The Help tab offers a quick reference to various fly commands.

8.2 Editing Window[s]

The editing window is a regular vim window. There could be several of those open based on a user's vim multi-window usage style. The plugin remembers current window of focus and a hit on the result line for a query initiated from that window opens the file in the same window. Buffer stacks are maintained on per window basis. Navigation (up and down) on a buffer stack could be done for each editing window separately.

9 Commands Description

9.1 Launch Command

9.1.1 SW (Switch)

This command is used to launch/display or hide the Fly control window. The first call initializes the plugin and shows the control window at the bottom. Subsequent calls just toggle the display of the control window. Context of fly control window is preserved across the toggles. The context includes settings, cache and results for each command executed so far.

9.2 Cscope Commands

9.2.1 BD (Build Cscope Databases)

This command builds cscope database for the source tree specified by `g:root`. If `g:root` is not specified the cscope database is built for the source tree rooted at current directory.

The database is built in background and the command returns immediately. You may start using Cscope search commands right away for smaller source trees as the build succeeds quickly. The build may take longer for larger code bases and the plugin will notify you in case you run queries before the database is ready.

The built database is kept in $CSCOPE_DIR if that is specified or else it is kept in the root directory of the source tree.

9.2.2 LD (List Cscope Databases)

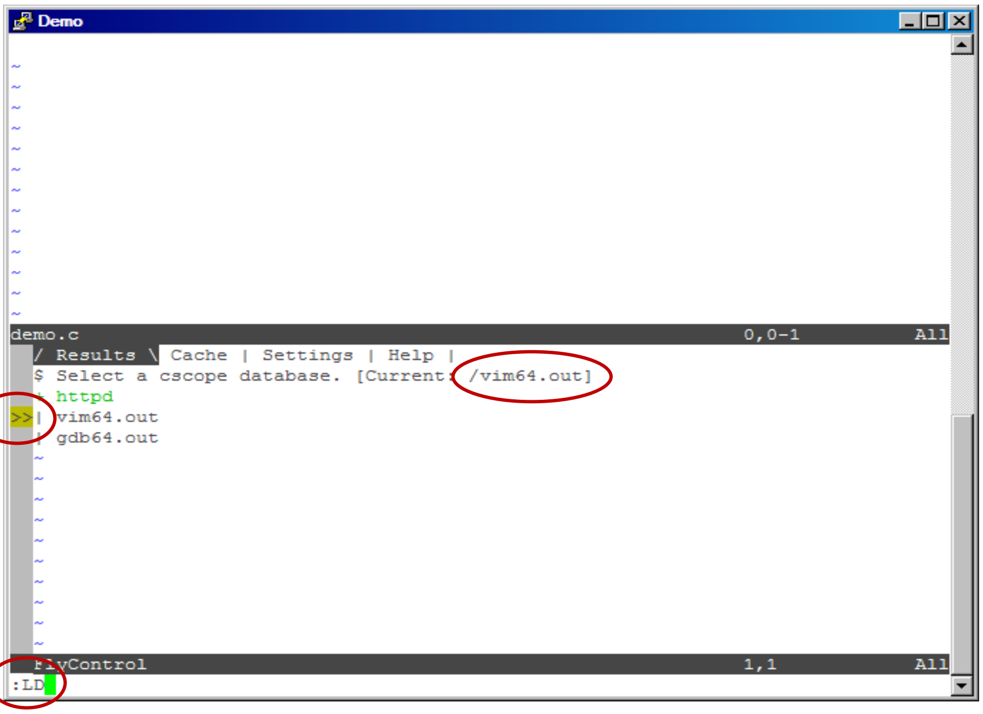
This command shows all the pre-built Cscope databases in $CSCOPE_DIR. If there are multiple Cscope databases for one or multiple sandboxes, they could be organized in a directory tree structure. The results will show the listing in explorer trees that could be collapsed or expanded.

A Cscope database could be selected by <Enter> on any of the result lines. The selected Cscope database shows up in the topmost line in result tab and the environment section in Settings tab.

An explorer tree for the selected Cscope database shows up on Cache Tab. Queries done on this Cscope database could be revisited by exploring this tree in Cache. Cscope database could be revisited by exploring this tree in Cache.

The Cscope database selection could be changed at any point of time.

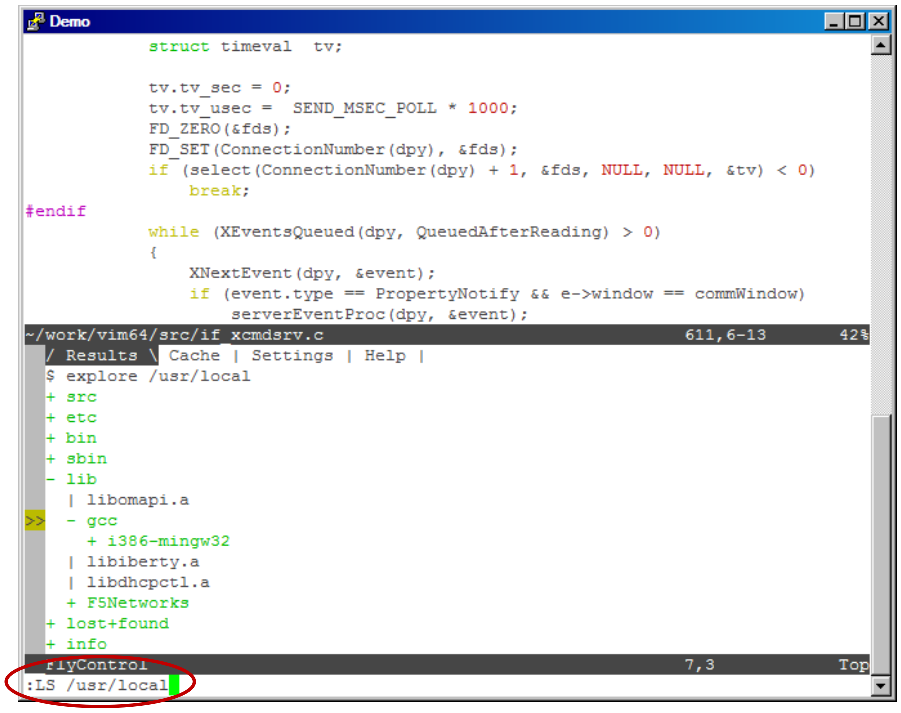
9.3 List Commands



```
~
~
~
~
~
~
~
~
~
~
~
~
~
demo.c                                                0,0-1          All
  / Results \ Cache | Settings | Help |
  $ Select a cscope database. [Current: /vim64.out]
  + httpd
>>| vim64.out
  | gdb64.out
  ~
  ~
  ~
  ~
  ~
  ~
  ~
  ~
  ~
  ~
  flyControl                                          1,1            All
:LD
```

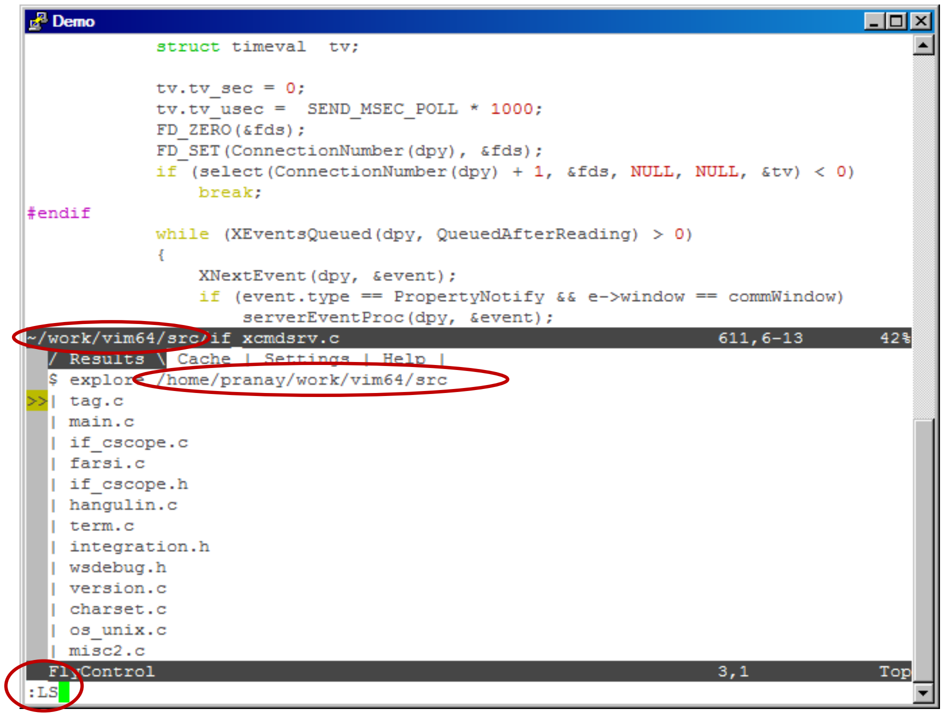
9.3.1 LS (List directory)

The LS command lists the directory specified in argument as an explorer-tree. User may open any of the files in result listing for editing just by <Enter> on the line listing the file.



```
            struct timeval  tv;

            tv.tv_sec = 0;
            tv.tv_usec =  SEND_MSEC_POLL * 1000;
            FD_ZERO(&fds);
            FD_SET(ConnectionNumber(dpy), &fds);
            if (select(ConnectionNumber(dpy) + 1, &fds, NULL, NULL, &tv) < 0)
                break;
#endif
            while (XEventsQueued(dpy, QueuedAfterReading) > 0)
            {
                XNextEvent(dpy, &event);
                if (event.type == PropertyNotify && e->window == commWindow)
                    serverEventProc(dpy, &event);
~/work/vim64/src/if_xcmdsrv.c                               611,6-13     42%
  / Results \ Cache | Settings | Help |
  $ explore /usr/local
  + src
  + etc
  + bin
  + sbin
  - lib
    | libomapi.a
>>  - gcc
      + i386-mingw32
    | libiberty.a
    | libdhcpctl.a
    + F5Networks
  + lost+found
  + info
  flyControl                                                7,3          Top
:LS /usr/local
```

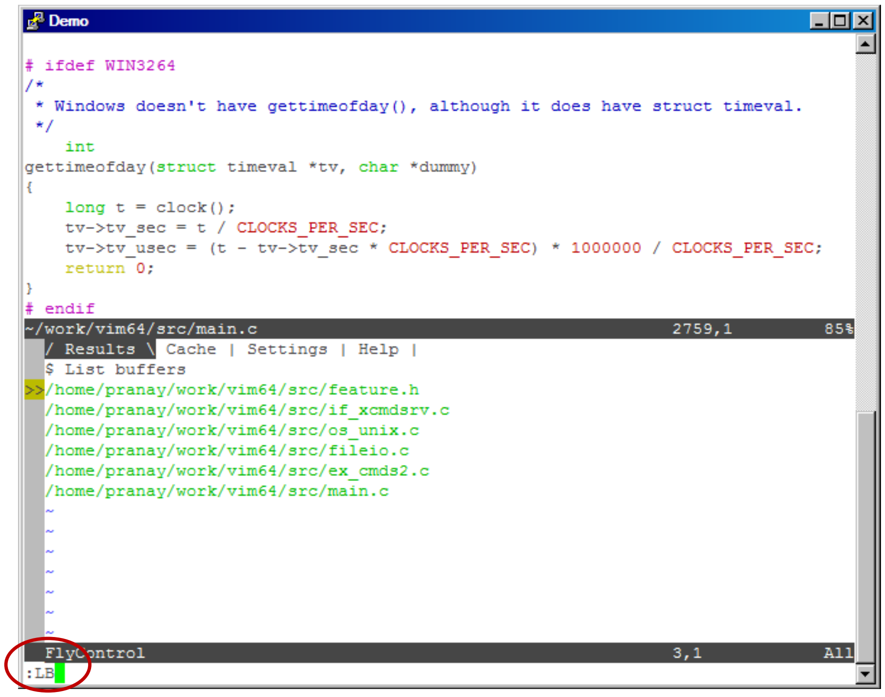
In case there are no arguments specified to LS, the directory of the file currently open in the window in focus is listed.



```
Demo
            struct timeval  tv;

            tv.tv_sec = 0;
            tv.tv_usec =  SEND_MSEC_POLL * 1000;
            FD_ZERO(&fds);
            FD_SET(ConnectionNumber(dpy), &fds);
            if (select(ConnectionNumber(dpy) + 1, &fds, NULL, NULL, &tv) < 0)
                break;
#endif
            while (XEventsQueued(dpy, QueuedAfterReading) > 0)
            {
                XNextEvent(dpy, &event);
                if (event.type == PropertyNotify && e->window == commWindow)
                    serverEventProc(dpy, &event);
~/work/vim64/src/if_xcmdsrv.c                               611,6-13        42%
  / Results \ Cache | Settings | Help |
  $ explore /home/pranay/work/vim64/src
>>| tag.c
  | main.c
  | if_cscope.c
  | farsi.c
  | if_cscope.h
  | hangulin.c
  | term.c
  | integration.h
  | wsdebug.h
  | version.c
  | charset.c
  | os_unix.c
  | misc2.c
  FlyControl                                                3,1             Top
:LS
```

9.3.2 LB (List Buffers)

The LB command is basically the *ls* command in vim. However, the buffer listing is shown in fly control window. A user can reopen the file just by hitting an <Enter> on the buffer listing.



```
Demo

# ifdef WIN3264
/*
 * Windows doesn't have gettimeofday(), although it does have struct timeval.
 */
    int
gettimeofday(struct timeval *tv, char *dummy)
{
    long t = clock();
    tv->tv_sec = t / CLOCKS_PER_SEC;
    tv->tv_usec = (t - tv->tv_sec * CLOCKS_PER_SEC) * 1000000 / CLOCKS_PER_SEC;
    return 0;
}
# endif
~/work/vim64/src/main.c                                 2759,1          85%
  / Results \ Cache | Settings | Help |
  $ List buffers
>>/home/pranay/work/vim64/src/feature.h
  /home/pranay/work/vim64/src/if_xcmdsrv.c
  /home/pranay/work/vim64/src/os_unix.c
  /home/pranay/work/vim64/src/fileio.c
  /home/pranay/work/vim64/src/ex_cmds2.c
  /home/pranay/work/vim64/src/main.c
  ~
  ~
  ~
  ~
  ~
  ~
  ~
  FlyControl                                            3,1             All
:LB
```

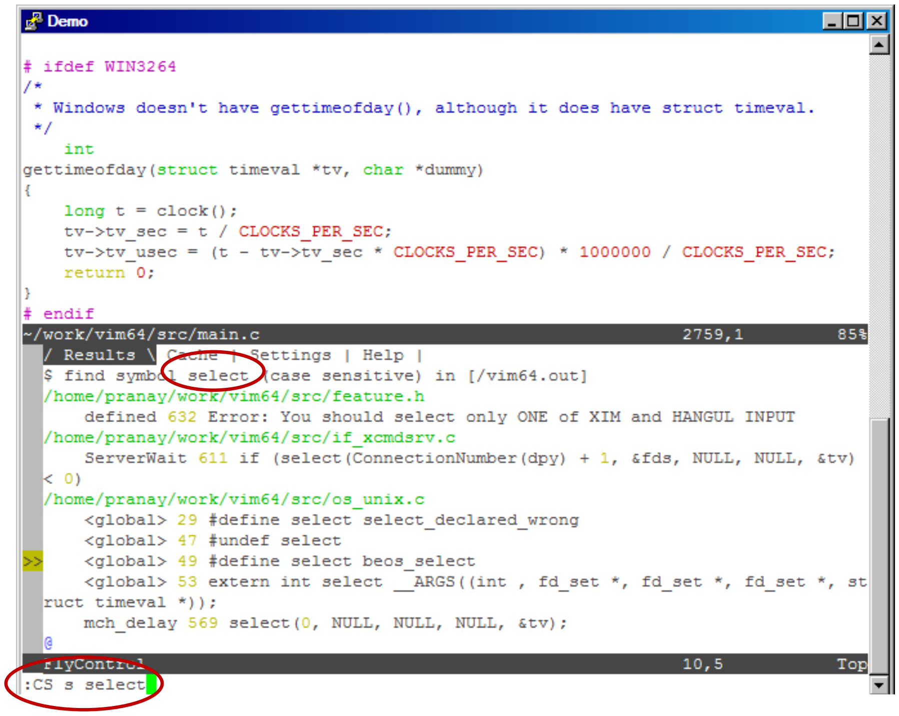
9.4 Symbol Search in Source Code

There are two commands offered for search, for now, - Cscope search and Grep.

9.4.1 CS (Query Cscope)

The CS commands perform traditional Cscope queries on currently selected Cscope database. The arguments to CS are the same characters that users of *cscope_maps.vim* are already familiar with (*cscope.sourceforge.net)*.

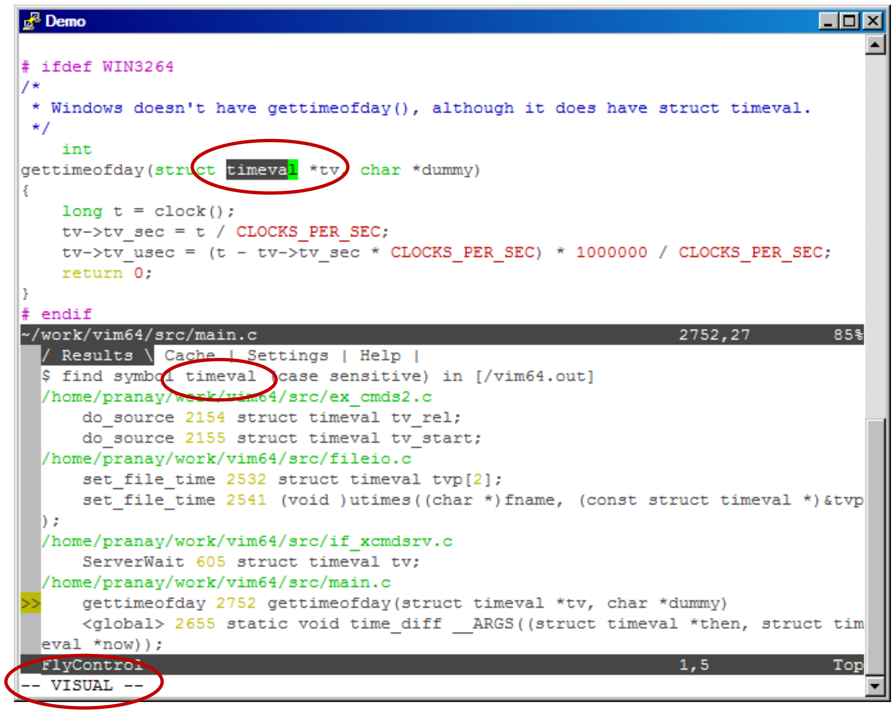
There are two *sets* of mappings defined for CS. The first set with <Leader><Leader> for performing queries on the *symbol under cursor*. The other set with <Leader> is for User to type in the symbol at vim command line. There is no auto-completion support yet for typing in these symbols. Only for file searches, partial name could be specified otherwise full symbol name is expected for the command.



```
# ifdef WIN3264
/*
 * Windows doesn't have gettimeofday(), although it does have struct timeval.
 */
    int
gettimeofday(struct timeval *tv, char *dummy)
{
    long t = clock();
    tv->tv_sec = t / CLOCKS_PER_SEC;
    tv->tv_usec = (t - tv->tv_sec * CLOCKS_PER_SEC) * 1000000 / CLOCKS_PER_SEC;
    return 0;
}
# endif
~/work/vim64/src/main.c                                      2759,1        85%
  / Results \ Cache | Settings | Help |
  $ find symbol select (case sensitive) in [/vim64.out]
  /home/pranay/work/vim64/src/feature.h
      defined 632 Error: You should select only ONE of XIM and HANGUL INPUT
  /home/pranay/work/vim64/src/if_xcmdsrv.c
      ServerWait 611 if (select(ConnectionNumber(dpy) + 1, &fds, NULL, NULL, &tv)
  < 0)
  /home/pranay/work/vim64/src/os_unix.c
      <global> 29 #define select select_declared_wrong
      <global> 47 #undef select
>>    <global> 49 #define select beos_select
      <global> 53 extern int select __ARGS((int , fd_set *, fd_set *, fd_set *, st
  ruct timeval *));
      mch_delay 569 select(0, NULL, NULL, NULL, &tv);
  @
  FlyControl                                                 10,5          Top
:CS s select
```

The very same sets of mappings work in visual mode too. First set <Leader><Leader> performs queries on the visually selected text. The second set <Leader> takes user to the command prompt but with visually selected text as argument and then user is free to change the text on command line.

Cscope queries can be run from any vim window.

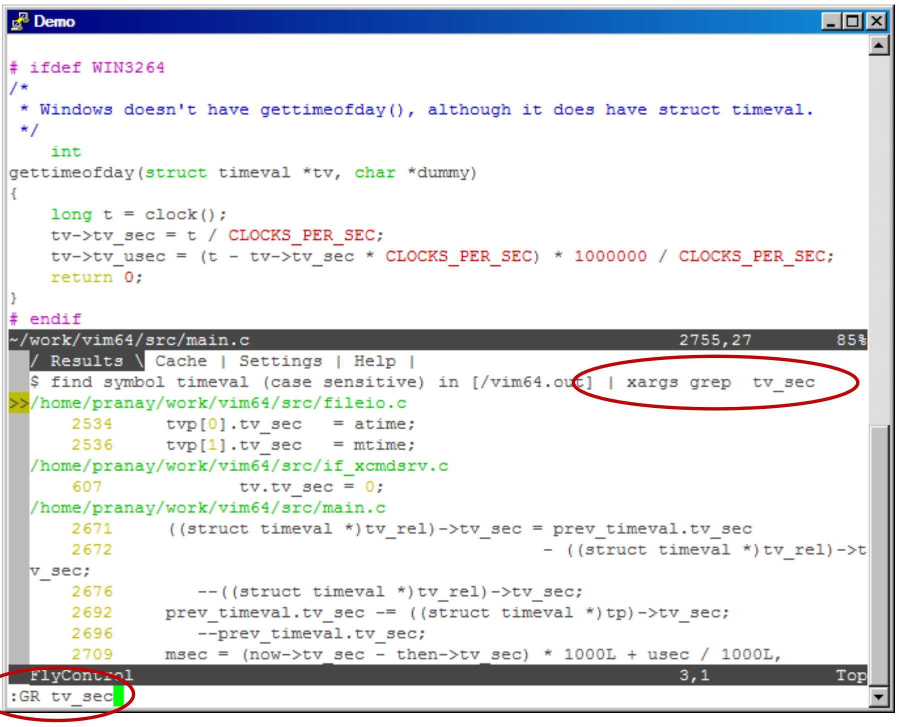


```
# ifdef WIN3264
/*
 * Windows doesn't have gettimeofday(), although it does have struct timeval.
 */
    int
gettimeofday(struct timeval *tv, char *dummy)
{
    long t = clock();
    tv->tv_sec = t / CLOCKS_PER_SEC;
    tv->tv_usec = (t - tv->tv_sec * CLOCKS_PER_SEC) * 1000000 / CLOCKS_PER_SEC;
    return 0;
}
# endif
~/work/vim64/src/main.c                                              2752,27        85%
  / Results \ Cache | Settings | Help |
  $ find symbol timeval (case sensitive) in [/vim64.out]
  /home/pranay/work/vim64/src/ex_cmds2.c
      do_source 2154 struct timeval tv_rel;
      do_source 2155 struct timeval tv_start;
  /home/pranay/work/vim64/src/fileio.c
      set_file_time 2532 struct timeval tvp[2];
      set_file_time 2541 (void )utimes((char *)fname, (const struct timeval *)&tvp
  );
  /home/pranay/work/vim64/src/if_xcmdsrv.c
      ServerWait 605 struct timeval tv;
  /home/pranay/work/vim64/src/main.c
>>    gettimeofday 2752 gettimeofday(struct timeval *tv, char *dummy)
      <global> 2655 static void time_diff __ARGS((struct timeval *then, struct tim
  eval *now));
  FlyControl                                                         1,5            Top
-- VISUAL --
```

9.4.2 GR (Grep)

The GR command is traditional UNIX [e]grep that runs on the set of files currently listed in fly control window (that may be as a result of another fly command e.g. LS, CS etc)

.



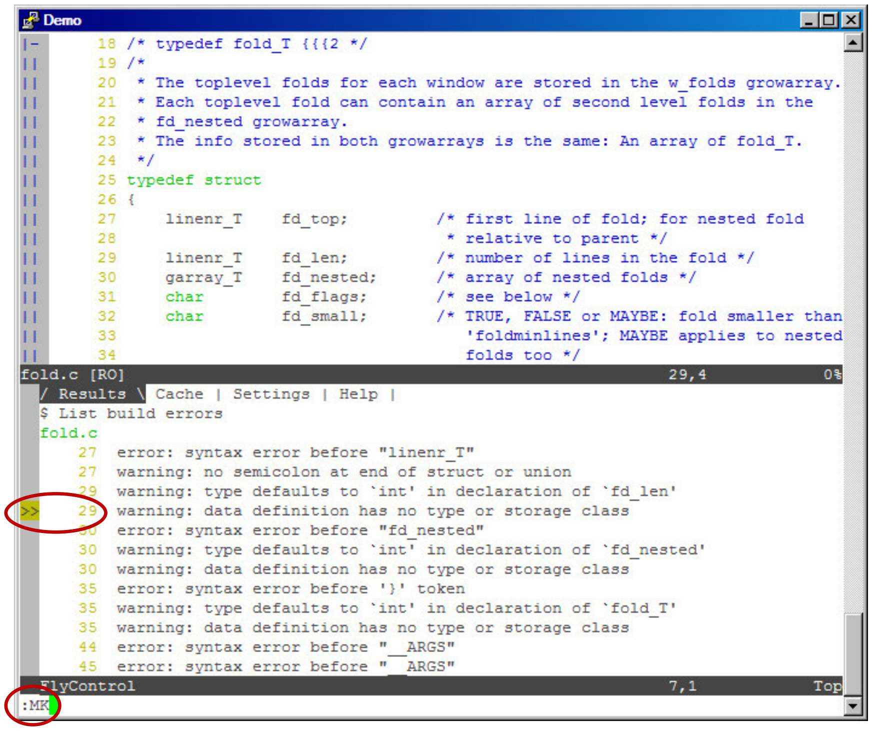
```
# ifdef WIN3264
/*
 * Windows doesn't have gettimeofday(), although it does have struct timeval.
 */
    int
gettimeofday(struct timeval *tv, char *dummy)
{
    long t = clock();
    tv->tv_sec = t / CLOCKS_PER_SEC;
    tv->tv_usec = (t - tv->tv_sec * CLOCKS_PER_SEC) * 1000000 / CLOCKS_PER_SEC;
    return 0;
}
# endif
~/work/vim64/src/main.c                                              2755,27        85%
  / Results \ Cache | Settings | Help |
  $ find symbol timeval (case sensitive) in [/vim64.out] | xargs grep  tv_sec
>>/home/pranay/work/vim64/src/fileio.c
      2534     tvp[0].tv_sec    = atime;
      2536     tvp[1].tv_sec    = mtime;
  /home/pranay/work/vim64/src/if_xcmdsrv.c
      607               tv.tv_sec = 0;
  /home/pranay/work/vim64/src/main.c
      2671     ((struct timeval *)tv_rel)->tv_sec = prev_timeval.tv_sec
      2672                                      - ((struct timeval *)tv_rel)->t
  v_sec;
      2676        --((struct timeval *)tv_rel)->tv_sec;
      2692     prev_timeval.tv_sec -= ((struct timeval *)tp)->tv_sec;
      2696        --prev_timeval.tv_sec;
      2709     msec = (now->tv_sec - then->tv_sec) * 1000L + usec / 1000L,
  FlyControl                                                         3,1            Top
:GR tv_sec
```

For GR, user may specify same options on vim command line for GR as he/she would for UNIX grep command on shell prompt.

9.5 Build Commands

9.5.1 MK (Make)

This command runs makeprg and shows results in the fly control window. And then user may open the errors/warnings by in edit windows. A user may visually highlight a phrase in an error message and perform an internet search on the selection. The error listings show up very slow if the error messages run in a few hundreds.



```
/* typedef fold_T {{{2 */
/*
 * The toplevel folds for each window are stored in the w_folds growarray.
 * Each toplevel fold can contain an array of second level folds in the
 * fd_nested growarray.
 * The info stored in both growarrays is the same: An array of fold_T.
 */
typedef struct
{
    linenr_T    fd_top;         /* first line of fold; for nested fold
                                 * relative to parent */
    linenr_T    fd_len;         /* number of lines in the fold */
    garray_T    fd_nested;      /* array of nested folds */
    char        fd_flags;       /* see below */
    char        fd_small;       /* TRUE, FALSE or MAYBE: fold smaller than
                                   'foldminlines'; MAYBE applies to nested
                                   folds too */
fold.c [RO]                                                 29,4        0%
 / Results \ Cache | Settings | Help |
 $ List build errors
 fold.c
     27  error: syntax error before "linenr_T"
     27  warning: no semicolon at end of struct or union
     29  warning: type defaults to `int' in declaration of `fd_len'
>>   29  warning: data definition has no type or storage class
     30  error: syntax error before "fd_nested"
     30  warning: type defaults to `int' in declaration of `fd_nested'
     30  warning: data definition has no type or storage class
     35  error: syntax error before '}' token
     35  warning: type defaults to `int' in declaration of `fold_T'
     35  warning: data definition has no type or storage class
     44  error: syntax error before "__ARGS"
     45  error: syntax error before "__ARGS"
 FlyControl                                                 7,1         Top
:MK
```

9.6 Web Commands

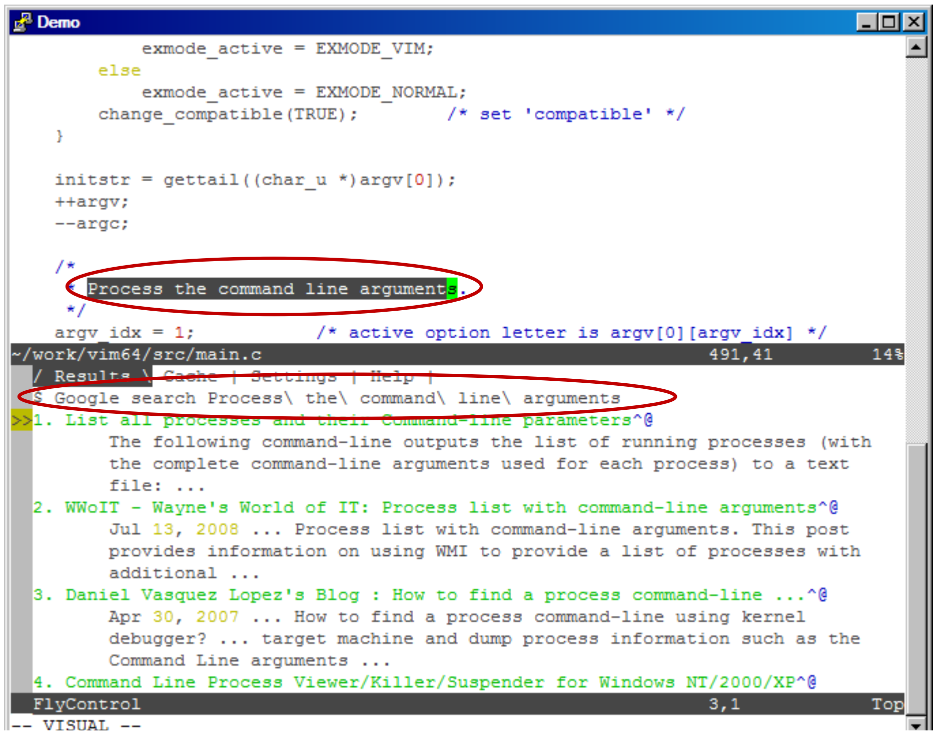
These commands are useful when one comes across a URL in the file being browsed or edited and wants to see the URL contents. These commands are also useful when you want to perform web search (on Google and Yahoo e.g.) for a phrase present in current file e.g. errors/warnings of compilation, usage of a library API etc.

Well, the URL contents are dumped in text format. And this is certainly not as effective as the real browsers – the intent here is to quickly browse web in the middle of code *flythrough*. It is useful to quickly cut and paste example code segments from web using vim commands or to quickly find out what an error or warning message may mean. This is not meant to be your web browser.

9.6.1 WS (<u>W</u>eb <u>S</u>earch)

This command performs web searches for text in search engines (Google or Yahoo).

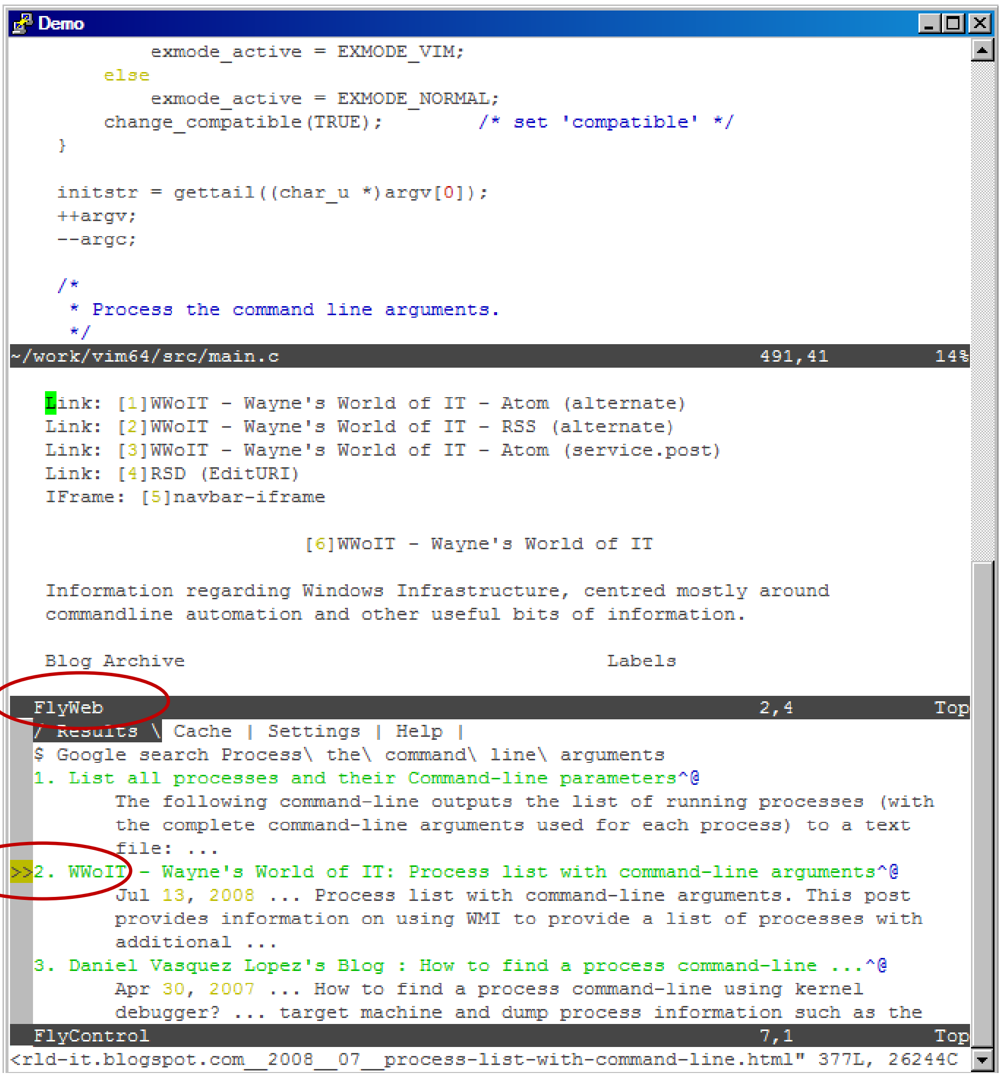
There are two kinds of mappings defined for WS. First one with <Leader><Leader> for performing web search for the *symbol under cursor* on currently selected Search Engine (Google or Yahoo). The other kind with <Leader> is for User to type in the text at vim command line.



```
        exmode_active = EXMODE_VIM;
    else
        exmode_active = EXMODE_NORMAL;
    change_compatible(TRUE);        /* set 'compatible' */
}

initstr = gettail((char_u *)argv[0]);
++argv;
--argc;

/*
 * Process the command line arguments.
 */
argv_idx = 1;               /* active option letter is argv[0][argv_idx] */
~/work/vim64/src/main.c                                        491,41        14%
 / Results \ Cache | Settings | Help |
 $ Google search Process\ the\ command\ line\ arguments
>>1. List all processes and their Command-line parameters^@
        The following command-line outputs the list of running processes (with
        the complete command-line arguments used for each process) to a text
        file: ...
  2. WWoIT - Wayne's World of IT: Process list with command-line arguments^@
        Jul 13, 2008 ... Process list with command-line arguments. This post
        provides information on using WMI to provide a list of processes with
        additional ...
  3. Daniel Vasquez Lopez's Blog : How to find a process command-line ...^@
        Apr 30, 2007 ... How to find a process command-line using kernel
        debugger? ... target machine and dump process information such as the
        Command Line arguments ...
  4. Command Line Process Viewer/Killer/Suspender for Windows NT/2000/XP^@
 FlyControl                                                    3,1           Top
-- VISUAL --
```

The very same mappings work in visual mode too. First kind with <Leader><Leader> performs web search on the visually highlighted text. The second one <Leader> takes user to the command prompt but with visually highlighted text as argument and then lets the user change the text on command line.

Results of the search are displayed in the result tab in fly control window. User may jump to any of the results by hitting the <Enter> key on that line – this results in contents being fetched from URL associated with the result line and the contents being displayed on a separate vim window (__FlyWeb) as shown in the next screenshot.



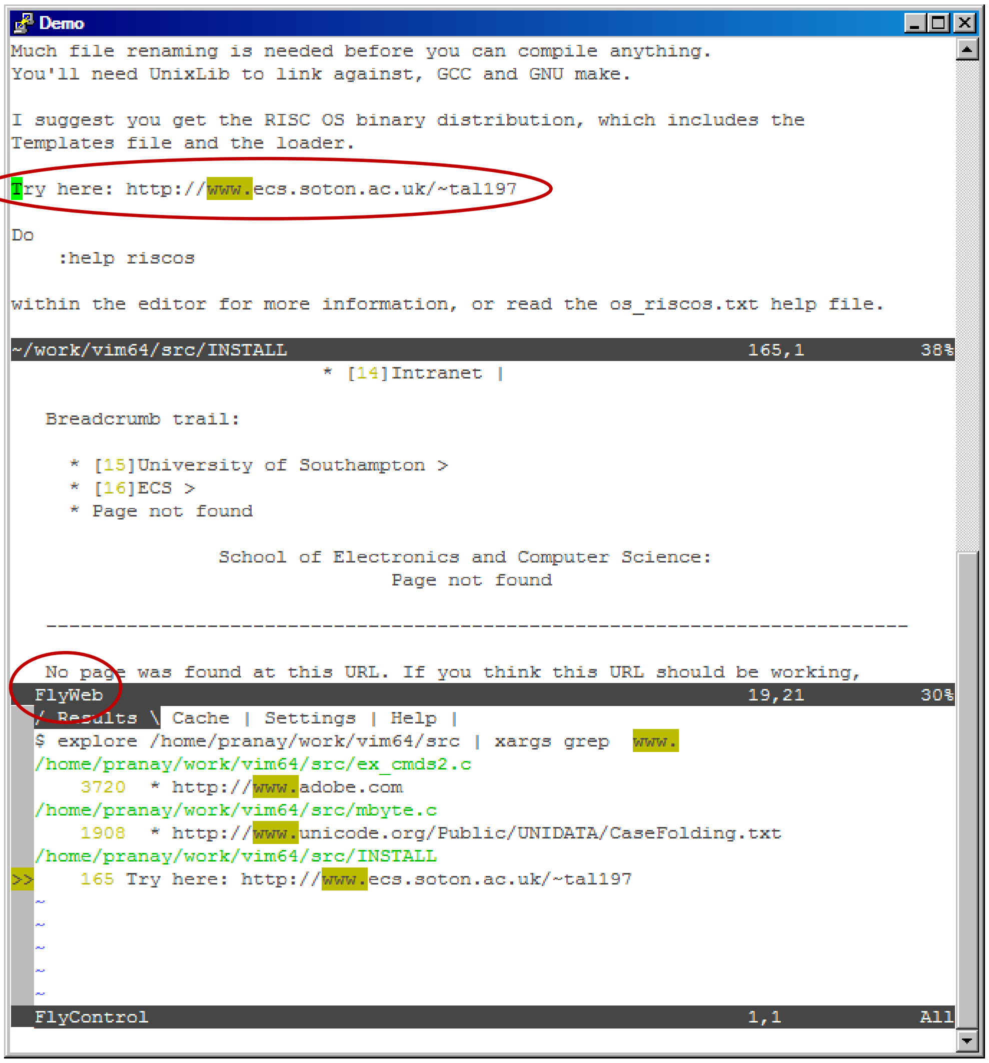
```
            exmode_active = EXMODE_VIM;
        else
            exmode_active = EXMODE_NORMAL;
        change_compatible(TRUE);        /* set 'compatible' */
    }

    initstr = gettail((char_u *)argv[0]);
    ++argv;
    --argc;

    /*
     * Process the command line arguments.
     */
~/work/vim64/src/main.c                                         491,41         14%

   Link: [1]WWoIT - Wayne's World of IT - Atom (alternate)
   Link: [2]WWoIT - Wayne's World of IT - RSS (alternate)
   Link: [3]WWoIT - Wayne's World of IT - Atom (service.post)
   Link: [4]RSD (EditURI)
   IFrame: [5]navbar-iframe

                         [6]WWoIT - Wayne's World of IT

   Information regarding Windows Infrastructure, centred mostly around
   commandline automation and other useful bits of information.

   Blog Archive                                    Labels

  FlyWeb                                                        2,4            Top
  / Results \ Cache | Settings | Help |
  $ Google search Process\ the\ command\ line\ arguments
  1. List all processes and their Command-line parameters^@
         The following command-line outputs the list of running processes (with
         the complete command-line arguments used for each process) to a text
         file: ...
>>2. WWoIT - Wayne's World of IT: Process list with command-line arguments^@
         Jul 13, 2008 ... Process list with command-line arguments. This post
         provides information on using WMI to provide a list of processes with
         additional ...
  3. Daniel Vasquez Lopez's Blog : How to find a process command-line ...^@
         Apr 30, 2007 ... How to find a process command-line using kernel
         debugger? ... target machine and dump process information such as the
  FlyControl                                                    7,1            Top
<rld-it.blogspot.com__2008__07__process-list-with-command-line.html" 377L, 26244C
```

9.6.2 OW (Open Web URL)

This command fetches contents of the page referenced by the URL argument and shows the contents in a separate vim window (__FlyWeb).

There are two kinds of mappings defined for OW too. First one with <Leader><Leader> for opening the URL that is present somewhere in the current line. The other one with <Leader> is for User to type in the URL on vim command line.

The very same mappings work in visual mode too. First one <Leader><Leader> opens the URL selected visually. The second one <Leader> takes user to the command prompt but with visually selected text as argument and then lets the user change text argument on vim command line.



```
Demo
Much file renaming is needed before you can compile anything.
You'll need UnixLib to link against, GCC and GNU make.

I suggest you get the RISC OS binary distribution, which includes the
Templates file and the loader.

Try here: http://www.ecs.soton.ac.uk/~tal197

Do
    :help riscos

within the editor for more information, or read the os_riscos.txt help file.

~/work/vim64/src/INSTALL                                      165,1          38%
                           * [14]Intranet |

   Breadcrumb trail:

     * [15]University of Southampton >
     * [16]ECS >
     * Page not found

                  School of Electronics and Computer Science:
                                 Page not found

   ---------------------------------------------------------------------------

   No page was found at this URL. If you think this URL should be working,
  FlyWeb                                                      19,21          30%
  / Results \ Cache | Settings | Help |
  $ explore /home/pranay/work/vim64/src | xargs grep  www.
  /home/pranay/work/vim64/src/ex_cmds2.c
      3720   * http://www.adobe.com
  /home/pranay/work/vim64/src/mbyte.c
      1908   * http://www.unicode.org/Public/UNIDATA/CaseFolding.txt
  /home/pranay/work/vim64/src/INSTALL
>>    165 Try here: http://www.ecs.soton.ac.uk/~tal197
  ~
  ~
  ~
  ~
  ~
  FlyControl                                                  1,1            All
```

9.7 Stack Commands

Concept of stack applies to editing windows (there may be multiple of those), web browsing window, man pages window and fly control window. The plugin maintains a stack for all windows currently open. User may navigate up and down on this stack in a window.

For edit windows the stack contains buffers/files that were opened in that window (e.g. user may start with opening a file in a window, but may move on to other files from the results of fly commands).

For the fly control window, the stack commands apply to the Results of previous fly commands. The stack navigation takes one back and forth through the results of fly commands executed so far.

For the web browsing window, the stack commands are equivalent to 'Back' and 'Forward' functions on a typical web browser.

For the man pages window, the stack commands helps one navigate through previously opened man pages. These are useful when you open the man page for a symbol and then follow the "SEE ALSO" section towards the end of a man page to see other man pages in the chain.

9.7.1 LF (Left)

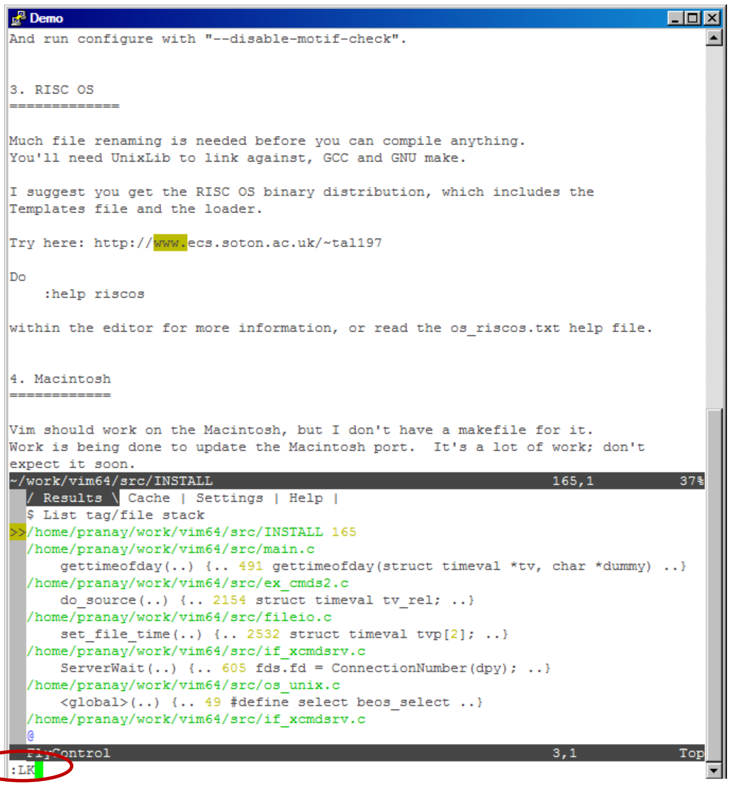
The LF command moves down the stack in current window of focus. The mapping for this command is <Leader>, (You may imagine using "<"– back/left key after <Leader>). Some users map <Ctrl-t> to this command. (Those who are used to *ctags*)

9.7.2 RT (Right)

The RT command moves up the stack in current window of focus. The mapping for this command is <Leader>. (You may imagine using ">"– forward/right key after <Leader>). Some users map <Ctrl-h> to this command.

9.7.3 LK (List Stack)

The LK command shows contents of stack in the current window of focus. This is useful when you are working on a new code base and want to take notes of the function call chain – just yank and paste the results of this command to your notes file.



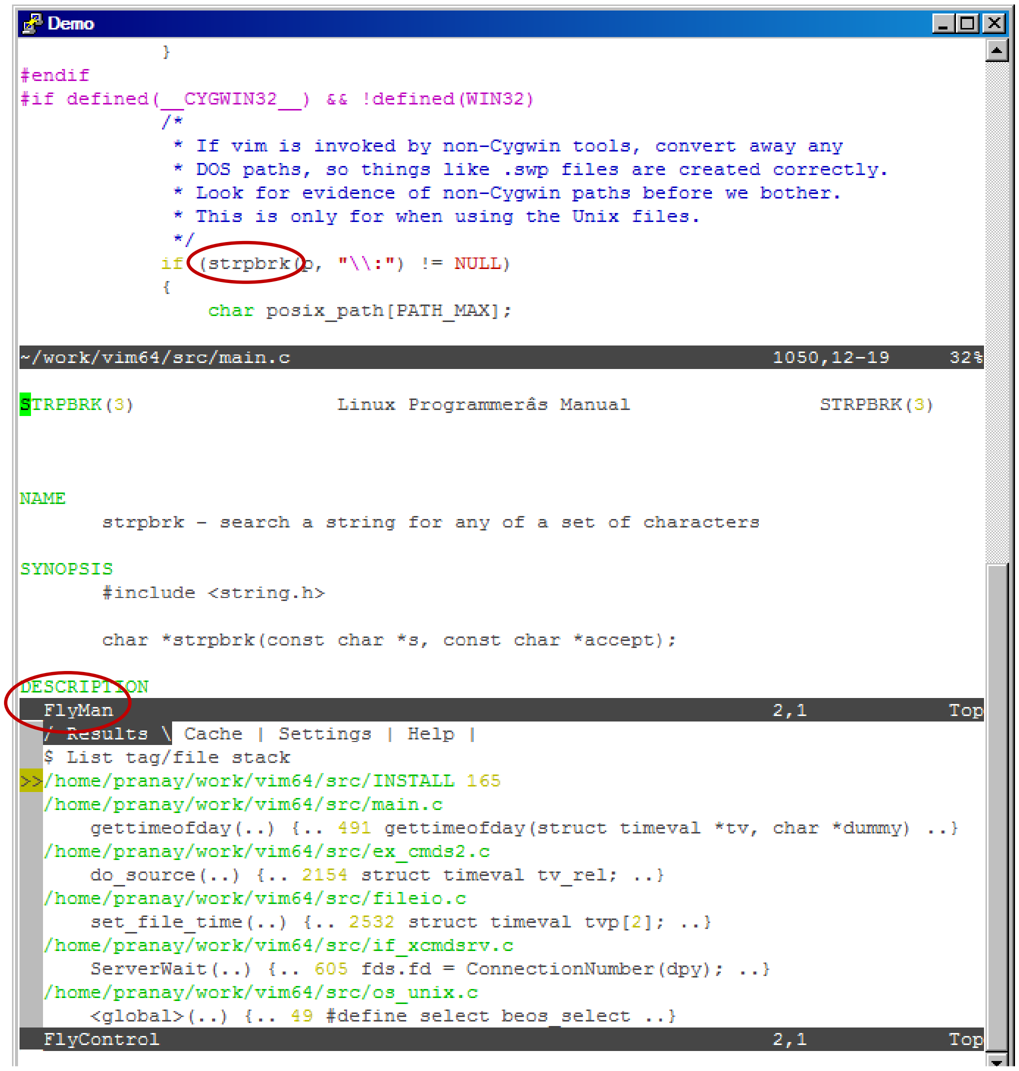
```
And run configure with "--disable-motif-check".


3. RISC OS
=============

Much file renaming is needed before you can compile anything.
You'll need UnixLib to link against, GCC and GNU make.

I suggest you get the RISC OS binary distribution, which includes the
Templates file and the loader.

Try here: http://www.ecs.soton.ac.uk/~tal197

Do
    :help riscos

within the editor for more information, or read the os_riscos.txt help file.


4. Macintosh
============

Vim should work on the Macintosh, but I don't have a makefile for it.
Work is being done to update the Macintosh port.  It's a lot of work; don't
expect it soon.
~/work/vim64/src/INSTALL                                      165,1          37%
  / Results \ Cache | Settings | Help |
  $ List tag/file stack
>>/home/pranay/work/vim64/src/INSTALL 165
  /home/pranay/work/vim64/src/main.c
      gettimeofday(..) {.. 491 gettimeofday(struct timeval *tv, char *dummy) ..}
  /home/pranay/work/vim64/src/ex_cmds2.c
      do_source(..) {.. 2154 struct timeval tv_rel; ..}
  /home/pranay/work/vim64/src/fileio.c
      set_file_time(..) {.. 2532 struct timeval tvp[2]; ..}
  /home/pranay/work/vim64/src/if_xcmdsrv.c
      ServerWait(..) {.. 605 fds.fd = ConnectionNumber(dpy); ..}
  /home/pranay/work/vim64/src/os_unix.c
      <global>(..) {.. 49 #define select beos_select ..}
  /home/pranay/work/vim64/src/if_xcmdsrv.c
  @
  Fly Control                                                 3,1            Top
:LK
```

9.8 UNIX Man Pages

The plugin allows users to open man pages in a vim window, thus allowing them to be able to view man pages when needed in an adjacent window, copy and paste *header file includes* and *example code segments* mentioned in the man pages.

Man page queries can be run from any window (including the man page window itself) and the results show up in __FlyMan window.

The man page window also has its own stack and thus one may navigate back and forth between previously opened man pages. The feature is useful when user wants to open man pages one after another following the chain of "*SEE ALSO*" section in man pages.



```
            }
#endif
#if defined(__CYGWIN32__) && !defined(WIN32)
            /*
             * If vim is invoked by non-Cygwin tools, convert away any
             * DOS paths, so things like .swp files are created correctly.
             * Look for evidence of non-Cygwin paths before we bother.
             * This is only for when using the Unix files.
             */
            if (strpbrk(p, "\\:") != NULL)
            {
                char posix_path[PATH_MAX];

~/work/vim64/src/main.c                                          1050,12-19     32%

STRPBRK(3)                  Linux Programmerâs Manual                STRPBRK(3)



NAME
       strpbrk - search a string for any of a set of characters

SYNOPSIS
       #include <string.h>

       char *strpbrk(const char *s, const char *accept);

DESCRIPTION
  FlyMan                                                         2,1            Top
  / Results \ Cache | Settings | Help |
  $ List tag/file stack
>>/home/pranay/work/vim64/src/INSTALL 165
  /home/pranay/work/vim64/src/main.c
      gettimeofday(..) {.. 491 gettimeofday(struct timeval *tv, char *dummy) ..}
  /home/pranay/work/vim64/src/ex_cmds2.c
      do_source(..) {.. 2154 struct timeval tv_rel; ..}
  /home/pranay/work/vim64/src/fileio.c
      set_file_time(..) {.. 2532 struct timeval tvp[2]; ..}
  /home/pranay/work/vim64/src/if_xcmdsrv.c
      ServerWait(..) {.. 605 fds.fd = ConnectionNumber(dpy); ..}
  /home/pranay/work/vim64/src/os_unix.c
      <global>(..) {.. 49 #define select beos_select ..}
  FlyControl                                                     2,1            Top
```

9.8.1 TM

Toggle display of Man Page window.

9.8.2 MAN <sym>

Open the default man page for the symbol (under cursor or typed).

9.8.3 MAN N <sym>

Open the man page for symbol (under cursor or typed) in section specified in argument.

9.9 Sandbox Comparison Commands

These commands are useful for comparing one sandbox to another sandbox or to the image in source code control repository. For example, before committing changes in a sandbox to source code repository, you may open list of modified files using RD command and then tie the sandbox to CVS using TCVS. You may then navigate through each of the modified file by opening them from the fly control window, the CVS diff is shown automatically in another window.

At least one file must have been opened in an edit window. Run these commands while you are (or the cursor is) in that edit window.

9.9.1 TCVS

This command ties your sandbox that is chosen automatically on choosing a cscope database to CVS. For every file that gets opened in the edit window using fly commands, CVS diff is shown in another window.

TCVS is a toggle command. The same command can be used to untie the sandbox from CVS.

The vim plugin *cvscommand.vim* must have been installed in order for this command to work.

9.9.2 TSVN

This command ties your sandbox that is chosen automatically on choosing a cscope database to SVN. For every file that gets opened in the edit window using fly commands, SVN diff is shown in another window.

TSVN is a toggle command. The same command can be used to untie the sandbox from SVN.

The vim plugin *svncommand.vim* must have been installed in order for this command to work.

9.9.3 TIE

This command ties your sandbox that is chosen automatically on choosing a cscope database to another sandbox that is specified using g:sandbox.

```
:let g:sandbox= "/absolute/path/to/root_of_sandbox/"
```

For every file that gets opened in the edit window using fly commands, vimdiff with corresponding file in the other sandbox is shown automatically in another window.

TIE is a toggle command. The same command can be used to untie the sandbox from the other one.

9.10 Restore Session

9.10.1 RS <file>

For now, it's implemented only for results of CS command. User may save the contents of fly control window (when results for a CS commands are displayed) in a file. That file could be opened later with RS command. Contents are displayed back in fly control window and user may jump to any of the result files.

9.11 Tab Navigation Commands

This is already covered in chapter 8.1.

9.12 Commands in 'Settings' Tab

This is already covered in chapter 8.1.3.

9.13 Misc

There are a few other commands like you could open any of the result files in a separate window other than the edit window from where fly command was run. The new window could be vertical or horizontally split.

You could also grep the results of CS/LS/GR command by editing the first line in result window to pipe it to UNIX grep command.

10 Known Issues

- Cscope database must be selected before using any of the CS commands or else behavior is undefined. The check is not put in the plugin before running CS.

- If the list of errors or warnings from MK were a few hundred in number, the MK command takes a couple of minutes to return.

- Error handling is not complete. The commands are expected to be used as documented or else sometimes you may see unexpected behavior.

- Sometimes, the buffer stack is lost.

In case you could reproduce a bug consistently. Please use “redir” command to capture the errors and report to the fly maintainers as follows.

```
:redir > reportfile.txt
```
Run your commands to reproduce the problem
```
:redir END
```
Send email to fly.vim@live.com with attachment reportfile.txt.

11 Wish list

I jotted down a few in my wish list below in no particular order. If you have more in mind you may add to the list and may be implement them into the plugin. Please send an email if you wish to do so to fly.vim@live.com.

- Multiple Cscope database selection. This would allow users to select multiple database and subsequent cscope queries will be run on each one of selected cscope databases.
- Support for GR command for results of most other commands.
- Support for delete commands (in results, cache).
- Update Cscope database[s] as the files are being edited.
- Function call graphs (as explorer trees).
- Implement cache for Web, Man and a few other commands.
- Auto completion of arguments. Currently the arguments to fly command are not automatically completed (if they were to be file names or code symbols present in database).
- Man page search. Currently the man pages are opened up for exact symbol. This search would allow user to search for man pages matching a symbol. (It would use `man -k` in background.
- Integrate a debugger (gdb).
- Build support for OpenGrok.
- Integrate GNATS/Bugzilla.

12 History and Future

The plugin was developed out of need to *filter* and *navigate* through a large number of Cscope search results back in vim 6.x days. Traditional Vim and Cscope integration was there but I didn't find it good enough to serve this need. Current state of this plugin has evolved to serve many more needs than this original one. But then, some of those needs are now natively served by vim 7.x+ also.

The best part of vim plugins is that most users are also developers. In larger user community the plugin may find more developers for itself. The plugin design is flexible to add new features. If there is enough user base and enough interest in contributing to development, I should probably put design guidelines (*Developer's Guide*) in public domain and may be put the plugin source in one of the open source hosting repositories.

In case you have any feedback or are interested in contributing to the development do send an email to fly.vim@live.com.

13 License

14 Disclaimer