Perl Hacks

on Vim

林佑安
Lin You-An
c9s / Cornelius
pause id: CORNELIUS

HI

I am 林佑安

Lin You-An

Taiwan

VIM & Perl

how can vim improve perl coding productivity

The worst way to edit

```
#!/usr/bin/env perl
my $hppiness = yapcasia->join();
```

```
#!/usr/bin/env perl
my $hppiness = yapcasia->join();
```

Oops!

lost “a”

```
#!/usr/bin/env perl
my $hppiness = yapcasia->join();
```

x N

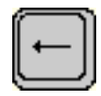
```
#!/usr/bin/env perl
my $hppiness = yapcasia->join();
```

x N

```
#!/usr/bin/env perl
my $hppiness = yapcasia->join();
```

x N

```
#!/usr/bin/env perl
my $hppiness = yapcasia->join();
```

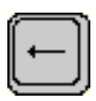


x N

```
#!/usr/bin/env perl
my $hppiness = yapcasia->join();
```

x N

```
#!/usr/bin/env perl
my $hppiness = yapcasia->join();
```

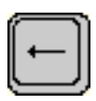


x N

```
#!/usr/bin/env perl
my $hppiness = yapcasia->join();
```

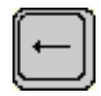
x N

```
#!/usr/bin/env perl
my $hppiness = yapcasia->join();
```



x N

```
#!/usr/bin/env perl
my $hppiness = yapcasia->join();
```



x N

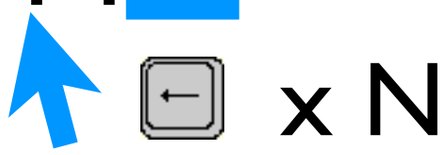
```
#!/usr/bin/env perl
my $hppiness = yapcasia->join();
```

x N

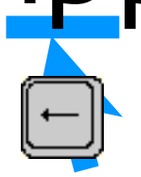
```
#!/usr/bin/env perl
my $hppiness = yapcasia->join();
```

x N

```
#!/usr/bin/env perl
my $hppiness = yapcasia->join();
```



```
#!/usr/bin/env perl
my $hppiness = yapcasia->join();
```



```
#!/usr/bin/env perl
my $hppiness = yapcasia->join();
```

```
#!/usr/bin/env perl
my $happiness = yapcasia->join();
```

```
#!/usr/bin/env perl
my $happiness = yapcasia->join();
```

if this costs 4 sec

```
#!/usr/bin/env perl
my $happiness = yapcasia->join();
```

if this costs 4 sec

if x 50 times this kind of situation per day

```
#!/usr/bin/env perl
my $happiness = yapcasia->join();
```

if this costs 4 sec

if x 50 times this kind of situation per day

```
#!/usr/bin/env perl
my $happiness = yapcasia->join();
```

if this costs 4 sec

if x 50 times this kind of situation per day

and you work for more than 300 day per year

= 16.6 hours

```
#!/usr/bin/env perl
my $happiness = yapcasia->join();
```

if this costs 4 sec

x 50 times this kind of situation per day ?

and you work for more than 300 day per year

= 16.6 hours

```
#!/usr/bin/env perl
my $happiness = yapcasia->join();
```

if this costs 4 sec

Awful

= 16.6 hours

What can you do in 16.6 hours

with family

with friends

kids

or more hacking

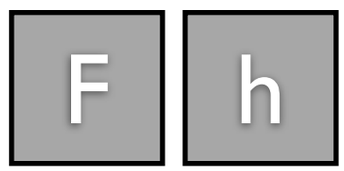
sleep...

anyway , time is money

The VIM way...

VIM:

```
#!/usr/bin/env perl
my $hppiness = yapcasia->join();
```



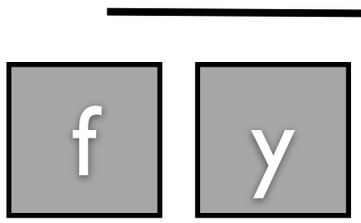
VIM:

```
#!/usr/bin/env perl
my $happiness = yapcasia->join();
```

a ppend a

VIM:

```
#!/usr/bin/env perl
my $happiness = yapcasia->join();
```



VIM:

```
#!/usr/bin/env perl
my $happiness = Yapcasia->join();
```

VIM:

```
#!/usr/bin/env perl
my $happiness = YApcasia->join();
```

VIM:

```
#!/usr/bin/env perl
my $happiness = YAPcasia->join();
```

VIM:

```
#!/usr/bin/env perl
my $happiness = YAPCasia->join();
```

So What is VIM ?

VIM is not an IDE

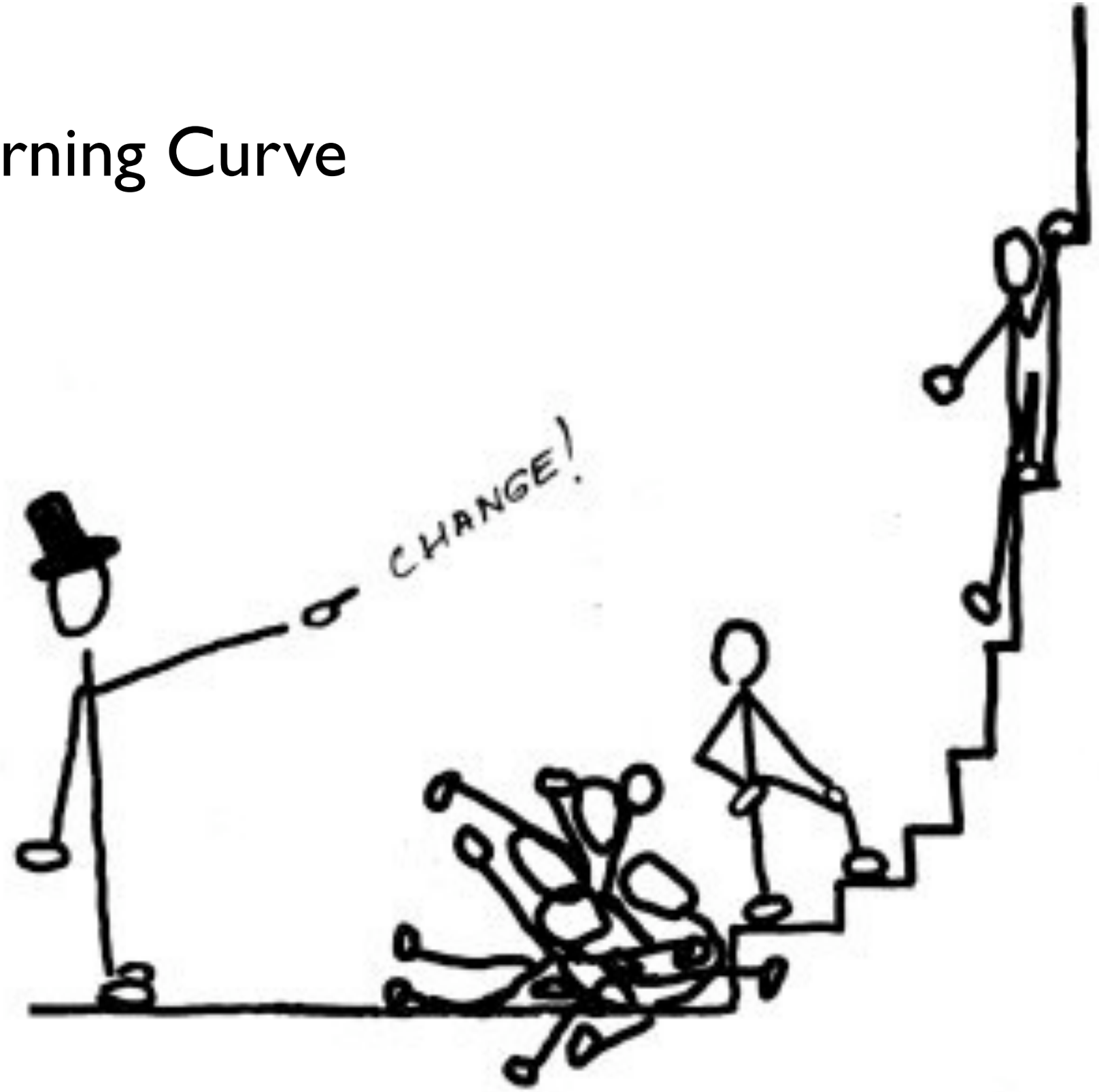
VIM is an

Editor

VI Improved

Move More Efficiently.

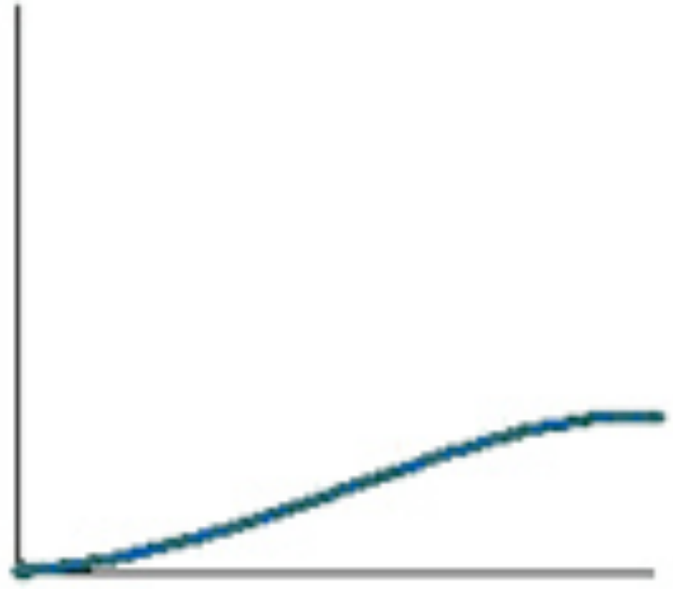
Learning Curve



Classical learning curves for some common editors

Notepad

Pico



Visual Studio

vi

emacs

Features

L Mode

More Than

4 Edit Mode

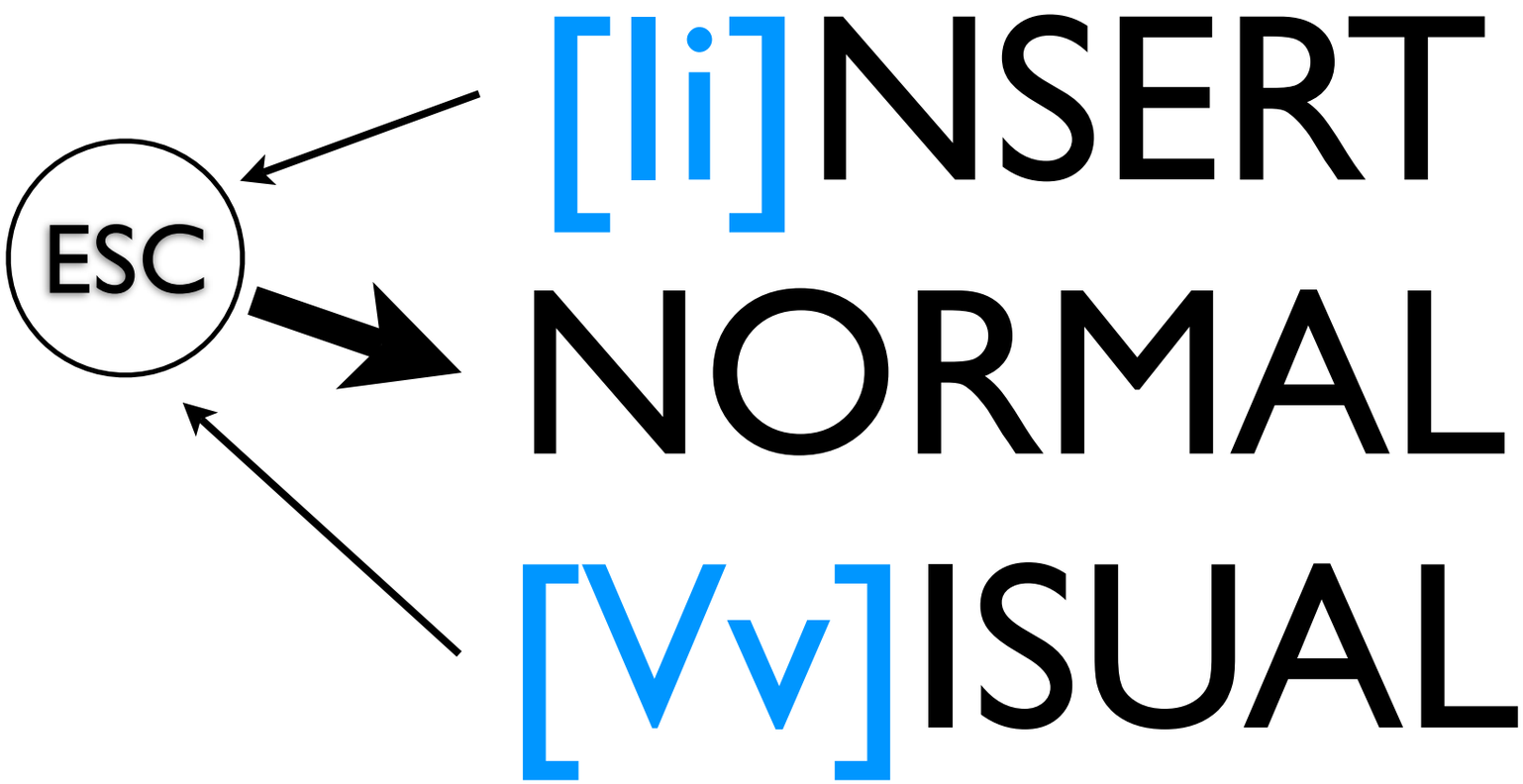
INSERT

NORMAL

VISUAL

SELECT

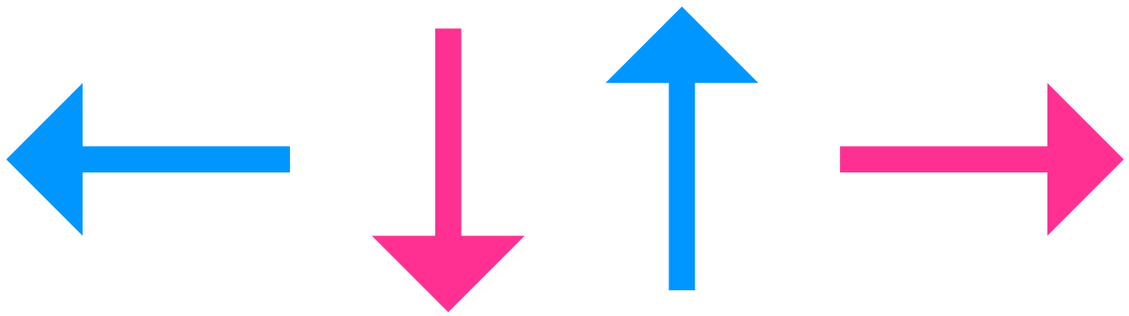
... et cetera


ESC
[i]NSERT
NORMAL
[Vv]ISUAL

1.1 Normal Mode

Motion

HJKL



h , j , k , l

H , M , L

w , e , b

f[x] , t[x]

[{ ,]} , %

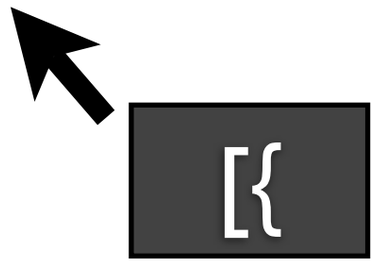
(,) , { , }

```
# comments ...

foreach my $foo ( @bar ) {

    # do something

    my @outs = grep /pattern/ , @list;

}

sub func1 {


}
```

```
# comments ...

foreach my $foo ( @bar ) {

    # do something

    my @outs = grep /pattern/ , @list;

}

sub func1 {


}
```

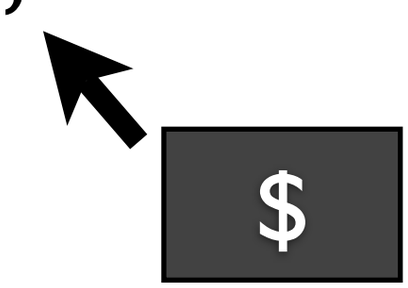
Cursor

```
# comments ...

foreach my $foo ( @bar ) {

    # do something

    my @outs = grep /pattern/ , @list;

}

sub func1 {


}
```

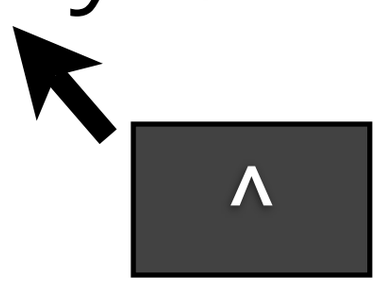


```
# comments ...

foreach my $foo ( @bar ) {

    # do something

    my @outs = grep /pattern/ , @list;

}

sub func1 {


}
```

```
# comments ...

foreach my $foo ( @bar ) {

    # do something

    my @outs = grep /pattern/ , @list;

}

sub func1 {


}
```



```
# comments ...

foreach my $foo ( @bar ) {

    # do something

    my @outs = grep /pattern/ , @list;
}

sub func1 {


}
```



H

画面最上方

```
# comments ...

foreach my $foo ( @bar ) {

    # do something

    my @outs = grep /pattern/ , @list;

}

sub func1 {


}
```

```
# comments ...

foreach my $foo ( @bar ) {

    # do something

    my @outs = grep /pattern/ , @list;

}

sub func1 {


}
```

M

画面中間行

```
# comments ...

foreach my $foo ( @bar ) {

    # do something

    my @outs = grep /pattern/ , @list;

}

sub func1 {


}
```

L

画面最下方

```
# comments ...

foreach my $foo ( @bar ) {

    # do something

    my @outs = grep /pattern/ , @list;

}

sub func1 {


}
```

修改至行尾

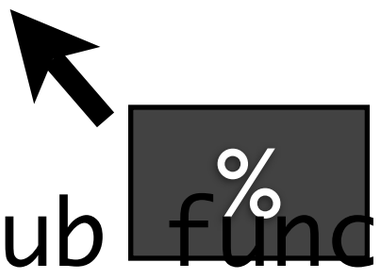
```
# comments ...

foreach my $foo ( @bar ) {

    # do something

    my @outs = grep /pattern/ , @list;

}

sub func1 {


}
```

C

並進入 Insert mode

```
# comments ...

foreach my $foo ( @bar ) {

    # do something

    my @outs = grep /pattern/ , @list;

}

sub func1 {


}
```

```
# comments ...

foreach my $foo ( @bar ) {

    # do something

    my @outs = grep /pattern/ , @list;


}

sub func1 {


}
```



:h motion.txt

1.2 Insert Mode

Editing text

Insert Mode

- i : Insert text before the cursor

Insert Mode

- i : Insert text before the cursor
- I : Insert text before the first non-blank in the line

Insert Mode

- i : Insert text before the cursor
- I : Insert text before the first non-blank in the line
- a : Append text after the cursor

Insert Mode

- i : Insert text before the cursor
- I : Insert text before the first non-blank in the line
- a : Append text after the cursor
- A : Append text at the end of the line

1.3 Visual Mode

Select region

Visual Mode

- v : start Visual mode per character.

Visual Mode

- v : start Visual mode per character.
- V : start Visual mode linewise.

Visual Mode

- v : start Visual mode per character.
- V : start Visual mode linewise.
- Ctrl-v : start Visual mode blockwise.

Visual Mode

- v : start Visual mode per character.
- V : start Visual mode linewise.
- Ctrl-v : start Visual mode blockwise.

operator 如 y (yank) , d (delete) , c (change) ... etc

2. Syntax Highlight Support

```
/opt/local/share/vim/vim72
$ ls -1 syntax/ | wc -l
     520
```

more than 500 syntax files

Customizable Syntax

Perl needs syntax

but we design more syntax in Perl

Modules need syntax too

Template::Declare

```
template simple => sub {
    html {
        head {}
        body {
            p {'Hello, world wide web!'}
        }
    }
};

 # ... templates
```

```
template 'index' => sub {

};

template 'foo' => sub {

};

template 'bar' => sub {

};

template 'yapc' => sub {

};
```

```
 10
 11 +---  3 lines: template 'index' => sub {----
 14
 15 +---  3 lines: template 'foo' => sub {------
 18
 19 +---  3 lines: template 'bar' => sub {------
 22
 23 +---  3 lines: template 'yapc' => sub {-----
 26
```

```
package MyView;
use Markapl;

tempalte '/a/page.html' => sub {
    h1("#title") { "Hi" };
    p(".first") { "In the begining, lorem ipsum...." };
    p(style => "color: red;") { "But...." };
}
```

Jifty::DBI::Schema

```
package TestApp::Model::Phone;
use Jifty::DBI::Schema;
use Jifty::DBI::Record schema {
    column user  =>
        references TestApp::Model::User by 'id',
        is mandatory;
    column type  => ...;
    column value => ...;
};
```

~/.vim/syntax/[filetype].vim

~/.vim/syntax/[filetype].vim

```
syn match [id] [re] [options]
```

~/.vim/syntax/[filetype].vim

```
syn match [id] [re] [options]
syn region [id] start=[re] end=[re]
```

~/.vim/syntax/[filetype].vim

```
syn match [id] [re] [options]
syn region [id] start=[re] end=[re]
syn keyword [id] [keyword] ...
```

~/.vim/syntax/[filetype].vim

```
syn match [id] [re] [options]
syn region [id] start=[re] end=[re]
syn keyword [id] [keyword] ...

hi [id]  guibg=[color] ctermfg=[color]
```

```
~/.vim/syntax/perl/*.vim
~/.vim/syntax/perl/jifty.vim
~/.vim/syntax/perl/template-declare.vim
~/.vim/syntax/perl/markapl.vim
```

```
" Fold Jifty Templates
syn region jifty_td_fold
      start=+^\(private\s\+\)\=template\s+
      end=+^};+ transparent fold keepend

" Fold Jifty Template Delcare Tags
syn region jifty_tag_fold
      start="^\z(\s*\)\<\(div\|table\|row\|form\|script\|
cell\|row\|th\|tfoot\)\>\s*{"
      end="^\z1};\=" fold keepend
```

Fold Jifty::DBI schema columns and action params

```
  syn region jfscolumn
        start=+^\s*\(param\|column\)\>+
        end=+;+ contained
        contains=jfspropertybe,jfsproperty,jfscol,jfscolname,
           \ perlComment, perlString, perlFunction, perlFloat,
perlNumber, perlSpecialString, perlStringUnexpanded
           \ fold
```

:help syntax.txt

4.

key mapping

```
:map
:nmap
:vmap
:imap
:smap
:xmap
```

... more

:map *(all)*

:nmap *(normal mode)*

:vmap *(visual mode)*

:imap *(insert mode)*

:smap *(select mode)*

:xmap *(visual , select mode)*

... more

```
:nmap <C-c><C-c> :!perl -Wc %
```

Ctrl C Ctrl C

let perl check current perl script syntax

use Tab and Shift-Tab to indent region ,
instead of “>” and “<“

```
nmap        <tab>    v>
nmap        <s-tab>  v<
```

```
vmap        <tab>    >gv
vmap        <s-tab>  <gv
```

```
imap <F2> <C-R>=strftime("%c")<CR>
imap <F2>  <C-R>=system('perl -MDateTime -e
DateTime->now')<CR>
```

to insert timestamp

```
cmap    <c-a>    <home>
cmap    <c-e>    <end>
cnoremap  <c-b>      <left>
cnoremap  <c-d>      <del>
cnoremap  <c-f>      <right>
cnoremap  <c-n>      <down>
cnoremap  <c-p>      <up>

cnoremap  <esc><c-b> <s-left>
cnoremap  <esc><c-f> <s-right>
```

support bash-like key mapping

:h map.txt

5.

Text Object

Text Object

- word
- string
- paragraph
- block

⟹ action
(yank,delete,change ...etc)

Operator Mapping

v | c | d i | a { | [| (| “ | ‘

visual
change
delete

Operator

Inner Object
An Object

Region
{ }
[ ]
( )
“ “
‘ ‘

va{

```
4 sub func1 {
5
6     for ( 1 .. 10 ) {
7
8     }
9
10 }
```


4 sub func1 {
5
6 for 1 10
7
8
9
10 }

sub(blah , blah)

ci(

sub( )

sub(new_args)

“Hello World”

di”

“”

“Hello New World”

Vim7.1 Comes with tag block region

<aaa> Hola </aaa>

vit

<aaa> Hola </aaa>

Visual Select

Alias

"dl"	delete character (alias: "x")	|dl|
"diw"	delete inner word	*diw*
"daw"	delete a word	*daw*
"diW"	delete inner WORD (see |WORD|)	*diW*
"daW"	delete a WORD (see |WORD|)	*daW*
"dd"	delete one line	|dd|
"dis"	delete inner sentence	*dis*
"das"	delete a sentence	*das*
"dib"	delete inner '(' ')' block	*dib*
"dab"	delete a '(' ')' block	*dab*
"dip"	delete inner paragraph	*dip*
"dap"	delete a paragraph	*dap*
"diB"	delete inner '{' '}' block	*diB*
"daB"	delete a '{' '}' block	

8.

FOLDS

FOLDIS

```
346
347 +---  5 lines: =head2 web----------------------------------------
352
353 +---  3 lines: sub web {-------------------------------------------
356
357 +---  5 lines: =head2 subs----------------------------------------
362
363 +---  3 lines: sub subs {------------------------------------------
366
367 +---  5 lines: =head2 bus-----------------------------------------
372
373 sub bus {
374     my $class = shift;
375     my %args = ( connect => 1, @_ );
376     if (not $PUB_SUB and $args{connect}) {
377         my @args;
378
379         my $backend = Jifty->config->framework('PubSub')->{'Backend'};
380         if ( $backend eq 'Memcached' ) {
381             require IO::Socket::INET;
382
383             # If there's a running memcached on the default port. this
384             if ( IO::Socket::INET->new('127.0.0.1:11211') ) {
385                 @args = ( Jifty->app_instance_id );
386             } else {
387                 $backend = 'JiftyDBI';
388             }
389         }
```

Fold Methods

Fold Methods

Syntax Fold

Fold Methods

Syntax Fold

```
:set foldmethod=syntax
```

set fold method as syntax , check out more options in:
$VIMRUNTIME/syntax/*.vim

Perl built-in syntax

/opt/local/share/vim/vim72/syntax/perl.vim

```
if exists("perl_want_scope_in_variables")
“  .....
if exists("perl_extended_vars")
“  .....
  if exists("perl_fold")
“  .....
```

you can enable those features in your .vimrc

```
let perl_fold = 1
let perl_extended_vars = 1
“ .... etc
```

for complex things like
@{${"foo"}}.

```
let  perl_include_pod = 1
let  perl_extended_vars = 1
let  perl_want_scope_in_variables = 1
let  perl_fold = 1
let  perl_fold_blocks = 1
```

for something like
$pack::var1

Fold Methods

- Syntax Fold
- Marker Fold

Fold Methods

Syntax Fold
Marker Fold

```
:set foldmethod=marker
```

fold region by markers , the default marker is “{{{“ ,“}}}”

```
#  fold this {{{
sub do_something {

    # a lot of work ...

}
#   }}}
```

```
#  fold this {{{
sub do_something {

    # a lot of work ...
    # foldlevel 2 {{{2
        # foldlevel 4 {{{4

        # }}}4
      # foldlevel here is 3
    # }}}2
}
#  }}}
```

```
 1 " vim:fdm=marker:sw=2:ts=2:et:fdl=0:
 2 "============================
 3 " Author:    Cornelius < cornelius.howl{at}gmail{dot}com >
 4 " Web:       http://c9s.blogspot.com
 5 " Date:      Wed 28 Mar 2007 04:37:26 AM CST
 6 +-- 39 lines: Usage:
45 "============================
46 +-- 17 lines: general settings
63 +-- 86 lines: options
149 +-- 20 lines: built-in color
169 +-- 12 lines: menu
181 +-- 16 lines: fold options
197 +-- 23 lines: foldexpr
220 +-- 70 lines: foldtext
290 +-- 10 lines: skeleton
300 +-- 14 lines: autocommand
314 +--147 lines: keymap
461 +-- 12 lines: commands
473 +--  8 lines: convert <arrow
481 +-- 42 lines: buffer sel by pattern
523 +-- 29 lines: blogger template
552 +-- 15 lines: inserting prefix
567 +--  7 lines: all space
574 +-- 31 lines: find file
605 +-- 16 lines: vimgrep
621 +-- 13 lines: quickfix window
```

Fold Methods

- Syntax Fold
- Marker Fold
- Indent Fold

Fold Methods

Syntax Fold
Marker Fold
Indent Fold

```
:set foldmethod=indent
```

use indent to fold

Fold Methods

- Syntax Fold
- Marker Fold
- Indent Fold
- Manual Fold

```
:set foldmethod=manual
```

create folds manually

```
autocmd  BufWinLeave *.*  silent mkview
autocmd  BufWinEnter *.*  silent loadview
```

利用 autocmd 加上 mkview , loadview 來儲存手動建立的折疊區塊，儲存的折疊會被放在 ~/.vim/view/ 裡頭。

Fold Methods

- Syntax Fold
- Marker Fold
- Indent Fold
- Manual Fold
- Expr Fold *(Custom Fold Function)*

```
:set foldexpr=MyFoldLevel(v:lnum)
```

customized fold function

Fold Methods

- Syntax Fold
- Marker Fold
- Indent Fold
- Manual Fold
- Expr Fold *(Custom Fold Function)*
- Diff Fold

za

zA

zm

zM

zr

zR

zj , zk

[z ,]z

za

za When on a closed fold: open it. When folds are nested, you may have to use "za" several times. When a count is given, that many closed folds are opened.
When on an open fold: close it and set 'foldenable'. This will only close one level, since using "za" again will open the fold. When a count is given that many folds will be closed (that's not the same as repeating "za" that many times).

zA

zA When on a closed fold: open it recursively.
When on an open fold: close it recursively and set

zm Fold more: Subtract one from 'foldlevel'. If 'foldlevel' was already zero nothing happens.
'foldenable' will be set.

zM

zM Close all folds: set 'foldlevel' to 0.
'foldenable' will be set.

:h folding

10. QuickFix

:grep

```
:grep [pattern] [filepath]
```

Result ⇛ QuickFix Window

```
59
60      /* Don't overwrite an existing error. This preserves the first
61       * error, which is the most significant. */
62      _cairo_status_set_error (&font_face->status, status);
63
64      return _cairo_error (status);
65  }
66
67  void
68  _cairo_font_face_init (cairo_font_face_t               *font_face,
69                         const cairo_font_face_backend_t *backend)
70  {
71      CAIRO_MUTEX_INITIALIZE ();
72
73      font_face->status = CAIRO_STATUS_SUCCESS;
74      CAIRO_REFERENCE_COUNT_INIT (&font_face->ref_count, 1);
75      font_face->backend = backend;
```

```
cairo-font-face.c                                                    59,0-1
 5 cairo-font-face.c|69| const cairo_font_face_backend_t *backend)
 6 cairo-font-face.c|81| * cairo_font_face_reference:
 7 cairo-font-face.c|82| * @font_face: a #cairo_font_face_t, (may be %NULL in which case this
 8 cairo-font-face.c|87| * cairo_font_face_destroy() is made.
 9 cairo-font-face.c|89| * The number of references to a #cairo_font_face_t can be get using
10 cairo-font-face.c|90| * cairo_font_face_get_reference_count().
11 cairo-font-face.c|92| * Return value: the referenced #cairo_font_face_t.
12 cairo-font-face.c|94| cairo_font_face_t *
13 cairo-font-face.c|95| cairo_font_face_reference (cairo_font_face_t *font_face)
14 cairo-font-face.c|109| slim_hidden_def (cairo_font_face_r
[Quickfix List]
```

QuickFix Window

```
:set grepprg=/path/to/grep
```

:copen

:cclose

:cnext

```
:cprevious
```

QuickFix Window Toggle

```
com! -bang -nargs=? QFix cal QFixToggle(<bang>0)
fu! QFixToggle(forced)
  if exists("g:qfix_win") && a:forced == 0
    cclose
    unlet g:qfix_win
  else
    copen 10
    let g:qfix_win = bufnr("$")
  en
endf
nn       <leader>q :QFix<cr>
```

nmap to “\q” , the default <leader> key is “\”

9.

Helpful Settings

Should we re-indent code manually ?

PerlTidy++

```
:set equalprg=perltidy
```

```
:set equalprg=perltidy
```

to re-format a region , set perltidy as your reformater

then press “=” , it will pass to perltidy

```
autocmd Filetype perl :set equalprg=perltidy
```

when filetype is perl
use autocmd to set equalprg

to reformat SQL ,
we have SQL::Beautify module

SQL::Beautify

```
$ cat bin/sql-beautify.pl
#!/usr/bin/env perl
use warnings;
use strict;
use SQL::Beautify;
local $/;
my $sql = SQL::Beautify->new( query => <STDIN> , spaces =>
4 , break => "\n" );
print $sql->beautify;
```

```
autocmd Filetype sql :set equalprg=sql-beautify.pl
```

command! -range SQLF :'<,'>!/Users/c9s/bin/sql-beautify.pl

HTML Entities

```
$ cat bin/html-entities.pl
use HTML::Entities;
for ( <STDIN> ) {
    print encode_entities( $_ ) . "\n";
}
```

```
command! -range Entities :'<,'>!/Users/c9s/bin/html-entities.pl
```

```
" <a href="">YAPC ASIA</a>
```

```
&quot; &lt;a href=&quot;&quot;&gt;YAPC ASIA&lt;/a&gt;
```

Morse Encoder

```
use Convert::Morse qw(as_ascii as_morse is_morsable);
print as_morse( $_ ) .“\n” for ( <STDIN> );
```

```
command! -range Morse :'<,'>!/Users/c9s/bin/morse-encode.pl
```

```
sub handler_start {
    my ( $kernel, $heap, $session ) = @_[ KERNEL, HEAP, SESSION ];
    print "Session ", $session->ID, " has started.\n";
    $heap->{count} = 0;
    $kernel->yield('increment');
}
```

```
... ..- -...  .... .- -. -.. .-.. . .-. ..--.- ... - .- .-. -  {
   -- -.--  -.--.  $-.- . .-. -. . .-.. --..--  $.... . .- .--. --..--  $... . ... ... .. --- -.  -.--.-  -...-  .--.-. ..--.- [ -.-
   .--. .-. .. -. -  .-..-. ... . ... ... .. --- -.  .-..-. --..--  $... . ... ... .. --- -. -.....- >.. -.. --..--  .-..-.  .... .- .
   $.... . .- .--. -....- >{-.-. --- ..- -. - } -...-  ----- -.-.-
   $-.- . .-. -. . .-.. -....- >-.-- .. . .-.. -.. -.--. .----. .. -. -.-. .-. . -- . -. - .----. -.--.- -.-.-
}
```

source code

traverse

Case:

```
use Data::Dumper;
```

we need to find Data/Dumper.pm in @INC
then type something like
$ vim /opt/local/..../site_perl/Data/Dumper.pm

```
fu! GetCursorModuleName()
  let cw = substitute( expand("<cWORD>") , '.\{-}\(\(\w\+\)\(::\w\+
\)*\).*$' , '\1' , '' )
  return cw
endfunction


fu! TranslateModuleName(n)
  return substitute( a:n , '::' , '/' , 'g' ) . '.pm'
endf
```

```
fu! GetPerlLibPaths()
  let out = system('perl -e ''print join "\n",@INC''')
  let paths = split( out , "\n" )
  return paths
endf
```

```
fu! FindModuleFileInPaths()
  let paths = GetPerlLibPaths()
  let fname = TranslateModuleName(   GetCursorModuleName()  )
  let methodname = GetMethodName()

  if TabGotoFile( 'lib/' .  fname , methodname )
    return
  endif

  for p in paths
    let fullpath = p . '/' . fname
    if TabGotoFile( fullpath , methodname )
      break
    endif
  endfor
endfunction
```

nmap <leader>fm :call FindModuleFileInPaths()<CR>

type “\fm” rather than type $ perldoc -m Module::Name

Install CPAN Module from <cWORD>

```
nmap <C-x><C-i>  :call InstallCPANModule()<CR>
function! InstallCPANModule()
  let l = getline('.')
  let cw = substitute( expand('<cWORD>') , ";$" , "" , "g" )
  let cw = substitute( cw , "['\"]" , "" , "g" )
  echo "Installing CPAN Module: " . cw . "\n"
  silent exec "!cpanp i " . cw . " >& /dev/null"
  echo "Done\n"
endfunction
```

Skeletons

```
au bufnewfile *.pl  0r ~/.vim/skeleton/template.pl
au bufnewfile *.pod    0r ~/.vim/skeleton/template.pod
au bufnewfile *.pm    0r ~/.vim/skeleton/template.pm
```

more than one skeleton for perl code

```
fu! ReadTemplate()
  let sname = input( "template type:" )
  exec '0r ~/.vim/skeleton/perl/' . sname .
'.pl'
endf
command! NewFromSkeleton :call ReadTemplate()
```

```
nm  cr= ^f=cf;
nm  cl= ^f=c^
```

Pod Helper

```
fu! PodHelperFunctionHeader()
  let subname = substitute( getline('.') , 'sub\s\+\(\w\+\)\s
\+.*$' , '\1' , "" )
  let lines = [
        \ '=head2 ' . subname ,
        \ '' ,
        \ '' ,
        \ '' ,
        \ '=cut'  ,
        \ '',
        \]
  for text in lines
    :call append( line('.') - 1 , text )
  endfor
  :call cursor( line('.') - len( lines ) + 2 , 1  )
endf
nmap <leader>pf  :call PodHelperFunctionHeader()<CR>
```

```
abbr _perlbin #!/usr/bin/env perl
abbr _s $self
```

detect features

```
if has('perl')

endif
```

```
if has(‘netbeans_intg’)

endif
```

crossplatform settings

```
if has('mac')

elseif has(‘win32’)

elseif has(‘unix’)

endif
```

if you compiled vim
with perl

```
function! WhitePearl()
perl << EOF
        VIM::Msg("pearls are nice for necklaces");
        VIM::Msg("rubys for rings");
        VIM::Msg("pythons for bags");
        VIM::Msg("tcls????");
EOF
endfunction
```

```
:perl $a=1
:perldo $_ = reverse($_);1
:perl VIM::Msg("hello")
:perl $line = $curbuf->Get(42)
```

```
:perl VIM::Msg("Text")                    # displays a message
:perl VIM::Msg("Error", "ErrorMsg")       # displays an error message
:perl VIM::Msg("remark", "Comment")       # displays a highlighted message
:perl VIM::SetOption("ai")                # sets a vim option
:perl $nbuf = VIM::Buffers()              # returns the number of buffers
:perl @buflist = VIM::Buffers()           # returns array of all buffers
:perl $mybuf = (VIM::Buffers('qq.c'))[0] # returns buffer object for '
:perl @winlist = VIM::Windows()           # returns array of all windows
:perl $nwin = VIM::Windows()              # returns the number of windows
:perl ($success, $v) = VIM::Eval('&path') # $v: option 'path', $succes
:perl ($success, $v) = VIM::Eval('&xyz')  # $v: '' and $success: 0
:perl $v = VIM::Eval('expand("<cfile>")') # expands <cfile>
:perl $curwin->SetHeight(10)              # sets the window height
:perl @pos = $curwin->Cursor()            # returns (row, col) array
:perl @pos = (10, 10)
:perl $curwin->Cursor(@pos)               # sets cursor to @pos
:perl $curwin->Cursor(10,10)              # sets cursor to row 10 col 10
```

II.

Plugins

perlprove.vim

parse test output to quickfix window

':make *t', or ':make t'

SnipMate

```
snippet cla class .. initialize .. end
    class ${1:`substitute(Filename(), '^.', '\u&', '')`}
        def initialize(${2:args})
            ${3}
        end

end
```

code snippet

DBExt.vim

Database extension ' support
Oracle, Sybase, MSSQL ,
MySQL, DBI etc..

xml.vim

xml related

FuzzyFinder.vim

Fuzzy/Partial pattern explorer

The_NERD_TREE.vim

Directory Tree Explorer

The NERD Commenter

Comment Helper

taglist.vim

透過 ctags 工具產生程式碼標籤
並可將 macro , function , variable 等資料整理出來

autocomplpop.vim

MRU

most recently used

bufexplorer.vim

Buffer Explorer

git-vim

vimrc

http://oulixe.us/_vimrc
http://oulixe.us/_gvimrc

The old way to install vim scripts

http://www.vim.org/index.php

nerd_commenter

CPAN Linux NewLinks PasteBin Work Tmp Do MailBox CG-WLBARGLV3 Plk MyExp

SPONSOR Vim development VOTE for features

the editor

BUY the Vim book HELP Uganda LEARN Vim

not logged in (login)

Google™ Custom Search

Search

Home
Advanced search

About Vim
Community
News
Sponsoring
Trivia
Documentation
Download

Scripts
Tips
My Account

Site Help

What is Vim?

Vim is a highly configurable text editor built to enable efficient text editing. It is an improved version of the vi editor distributed with most UNIX systems. Vim is distributed free as charityware. If you find Vim a useful addition to your life please consider helping needy children in Uganda.

News

Vim 7.2.245 is the current version

Visit to Kibaale

[2009-05-12] Vim comes for free. I do ask you to consider helping Vim's charity, please read on. In April I have visited the Kibaale Childrens Centre. You can read my report, with lots of pictures, here. The new clinic has become very popular. Every day many patients come here for medical help that would otherwise not be available to them. The Kibaale clinic saves lives! This popularity comes at a price. We struggle to pay for all the medicine, salaries of the medical staff, maintenance of equipment, etc. Therefore I have set up this page for sponsoring a room in the clinic. We need your help! (*Bram Moolenaar*)

Vim 2009 desktop calendar

[2008-12-21] The usual Vim desktop calendar is now available for download and printing: http://www.moolenaar.net/#Calendar. If you print it on thick paper you can fold it so that it stands on your desk. One side contains a useful 12-month calendar. On the other side there is brief information about ICCF-Holland, Vim and A-A-P. Happy Vimming in 2009! (*Bram Moolenaar*)

more news... Get a Vim poster DVD and video about Vim's charity project

Recent Script Updates

2,721 scripts, 3,524,650 downloads

[2009-08-13] textobj-entire : Text objects for entire buffer
(0.0.1) - Fix a typo on the name of a variable. - *Kana Natsuno*
[2009-08-12] WuYe : A dark background color scheme
(1.2.1) The final version - *SAM Lee*
[2009-08-12] Screen (vim + gnu screen) : Start gnu screen w/ your vim session and a split shell + send commands to the sh
(0.4) - fixed usage in cygwin, which the previous version broke - fixed handling of large amounts of text sent through :ScreenSend (thanks to Tobias Wolf for

Search for Vim Scripts

on the web and the Vim website:

Google Code Search

You can use a regexp pattern, e.g., go{2}gle, h[ea]llo. (more about regexp patterns)

on the Vim website only

keywords nerd

type -- all --

sort by Rating Descending search cancel

Search Results

Searched scripts for "**nerd**"

Showing **1** to **3** of **3** results

Script	Type	Rating	Down loads	Summary
The NERD tree	utility	2095	24700	A tree explorer plugin for navigating the filesystem
Source Explorer (srcexpl.vim)	utility	903	4145	A Source code Explorer based on tags works like context window in Source Insight
trinity.vim	utility	98	1480	Build the trinity of srcexpl, taglist, NERD_tree to be a good IDE

prev | next

Showing **1** to **3** of **3** results

The NERD tree : A tree explorer plugin for navigating the filesystem

script karma	Rating **2107/596**, Downloaded by 24786

created by
Marty Grenfell

script type
utility

description
Grab the latest dev version from github: https://github.com/scrooloose/nerdtree.

What is this "NERD tree"??

Check out this demo http://www.flickr.com/photos/30496122@N07/2862367534/sizes

The NERD tree allows you to explore your filesystem and to open files and directories. It presents the filesystem to you in the form of a tree which you

package	script version	date	Vim version	user	release notes
NERD_tree.zip	**3.1.1**	*2009-06-07*	7.0	*Marty Grenfell*	- fix a bug where a non-listed no-name buffer was getting created every time the tree window was created, thanks to Derek Wyatt and owen1 - make <CR> behave the same as the 'o' mapping - some helptag fixes in the doc, thanks strull - fix an error when using :set nohidden and opening a file where the previous buf was modified. Thanks iElectric

Download

install details
NOTE: In version 2.0.0 the script file and help file were renamed to NERD_commenter.vim and NERD_commenter.txt.
If you are upgrading from version < 2.0.0 to version >= 2.0.0 then you must delete the old files NERD_comments.vim and
NERD_comments.txt.

Stick the NERD_comments.vim in ~/.vim/plugin and NERD_comments.txt in ~/.vim/doc.

Run :helptags ~/.vim/doc.

Restart vim.

Go :help NERD_commenter.txt to read the help file.

Read Install details

install details
NOTE: In version 2.0.0 the script file and help file were renamed to NERD_commenter.vim and NERD_commenter.txt.
If you are upgrading from version < 2.0.0 to version >= 2.0.0 then you must delete the old files NERD_comments.vim and
NERD_comments.txt.

Stick the NERD_comments.vim in ~/.vim/plugin and NERD_comments.txt in ~/.vim/doc.

Run :helptags ~/.vim/doc.

Restart vim.

Go :help NERD_commenter.txt to read the help file.

Read Install details

Awful!

Awful!

Vimana

Vim script Manager

- Vimball
- Archive File (zip , rar)
- .vim extension file

- Cache::File
- App::CLI
- Archive::Any
- ...etc

it's on CPAN

```
$ cpan Vimana
```

```
$ vimana search xml
```

```
$ vimana search xml
rrd.vim                - Edit RRD data with Vim.
qt.vim                 - tiny tool for the uic used in Qt from
Trolltech
syntax-for-xul         - Highlighting for XML User interface Language.
maven2.vim             - Compiler plugin for maven2
 .... skip
```

```
$ vimana info xml.vim
```

```
$ vimana install xml.vim
```

```
$ vimana install xml.vim
$ vimana install rails.vim
$ vimana install the-nerd-tree.vim
$ vimana install taglist.vim
$ vimana install snipmate
$ vimana install fuzzyfinder.vim
etc ...
```

ALL Works

Future Plan

- upgrade
- remove
- config
- support makefile

Git Repository

http://github.com/c9s/Vimana/tree/master

ENJOY

Thank You