

Tutorial Vim

@kholidfu*

https://github.com/kholidfu/vim_docs

August 3, 2018

Contents

1	Pendahuluan	3
1.1	Tentang Tutorial Ini	3
1.2	Tentang Vim	3
2	Penyuntingan Dasar	4
3	Navigasi	7
4	Fungsi Penyuntingan	10
4.1	Menyisipkan Teks	10
4.2	Salin dan Tempel	11
4.3	Sorot / Visual Mode	12
4.4	Fungsi Hapus / Potong	16
4.5	Undo dan Redo	16
4.6	Word Completion	16
4.7	Filename Completion	17
4.8	Replace Karakter	18
4.9	Menyunting Kata dalam tanda ' " (.	19
5	Fungsi Pencarian	19
5.1	Pencarian Case Insensitive	19
5.2	Pencarian Case Sensitive	20
5.3	Fungsi Cari dan Ganti	21
6	Macro	23
6.1	Dasar Macro	23
6.2	Melakukan Penomoran Secara Otomatis	24
7	Multi-tab	24

*<http://twitter.com/kholidfu>

8	Registers	25
9	Marks	26
10	Buffer	27
11	Code Folding	28
12	File Explorer	31
13	Split Screen	33
14	Session	34
15	Konfigurasi Vim	34
16	Lain-lain	35
16.1	Menjalankan Perintah Shell dari Vim	36
16.2	Menyisipkan Keluaran dari Shell Command ke dalam Vim . . .	36
16.3	Singkatan	37
16.4	Membatasi Panjang Baris Maksimum n Karakter	37
16.5	Menggabung Dua atau Lebih Baris	37
16.6	Lowercase, Uppercase dan Titlecase	38
16.7	Cari dan Hapus Baris Berdasar Pola	39
16.8	Membuat Baris Baru Identik	39
16.9	Cara Lain Beralih ke Command Mode	40
16.10	Bracket Matching	40
16.11	Mengaktifkan Penomoran Baris	41
16.12	Mengulang Perintah Terakhir	41
16.13	Eksekusi Kode Bash dalam Vim	41
16.14	Mengetahui Nama Berkas yang sedang disunting	42
16.15	Resize Splits	42

1 Pendahuluan

1.1 Tentang Tutorial Ini

Tulisan-tulisan ini sebenarnya bukanlah tutorial lengkap yang mengajarkan kepada Anda seluk-beluk program Vim, melainkan sekedar catatan pribadi penulis yang coba dituangkan kedalam sebuah berkas elektronik dengan tujuan untuk dokumentasi pribadi, syukur-syukur kalau ada pihak lain yang membacanya dan mampu mendapatkan manfaat dari tulisan-tulisan ini.

1.2 Tentang Vim

Vim adalah sebuah program penyunting teks yang dapat dikonfigurasi sesuai kebutuhan sehingga proses penyuntingan teks menjadi efisien. Vim merupakan versi vi yang terbaru dan disebarluaskan pada hampir semua sistem UNIX.¹

Vim sering dijuluki sebagai program 'penyunting bagi *programmer*', dan memang kenyataannya sangat berguna untuk kebutuhan *programming*, sehingga beberapa menyebutnya sebagai sebuah IDE (*Integrated Development Environment*). Kenyataannya, Vim tidak hanya digunakan para *programmer*. Vim sesuai digunakan untuk semua kebutuhan yang berkaitan dengan penyuntingan teks, dari menulis *email* sampai penyuntingan berkas konfigurasi.

Vim bukanlah program pemroses kata (*word processor*), dan tidak memiliki fitur WYSIWYG, Vim adalah sebuah alat penyunting yang harus dipelajari cara penggunaannya. Pertanyaan selanjutnya adalah sesulit apa sih belajar Vim itu? Sejauh pengetahuan penulis mempelajari Vim itu tidak akan ada habisnya, karena tujuan utama kita adalah efisiensi. Sesuatu yang hari ini dirasa sudah efisien belum tentu esok masih tetap efisien. Ini juga berarti proses *continuous improvement* berjalan terus-menerus. Hal ini juga berkaitan dengan seberapa kreatif si pengguna dalam mengoptimalkan Vim.

Vim adalah *charityware*, dan memiliki lisensi *GPL-compatible*, sehingga dapat didistribusikan secara bebas dan gratis, namun jika Anda merasa mendapatkan manfaat dari Vim, para pengembang Vim meminta Anda untuk membantu dengan melakukan donasi dengan membantu anak-anak di Uganda melalui ICCF.²

Perlu dicatat bahwa penggunaan Vim akan optimal dalam kondisi *mouseless operation*, artinya penggunaan perangkat *mouse* hanya akan memperlambat kerja kita, untuk itu mari kita biasakan bekerja tanpa *mouse*. Ini juga menjadi salah satu alasan kenapa banyak yang bilang belajar Vim itu sulit. Namun jangan khawatir, untuk bisa produktif dan efisien, Anda tidak perlu menguasai semua konsep Vim, cukup kuasai beberapa konsep dasar, setelah itu konsep lain dapat Anda pelajari sesuai kebutuhan.

¹<http://www.vim.org>

²<http://iccf-holland.org/>

2 Penyuntingan Dasar

Sebelum menjalankan Vim, tentunya Anda harus *install* dulu programnya, jika Anda menggunakan sistem operasi Ubuntu, Anda dapat meng-*install* menggunakan perintah:³

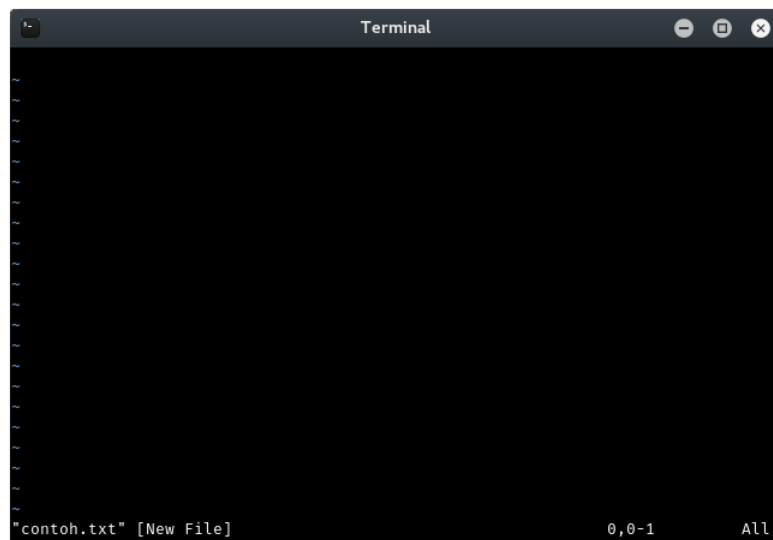
```
$ sudo apt install vim
```

Setelah Vim ter-*install*, hal penting dan mendasar sebenarnya adalah membuat berkas `.vimrc` pada direktori `home` Anda. Berkas ini berguna untuk konfigurasi program Vim, namun berhubungan materi tersebut masuk pada tingkatan menengah, mari kita jalankan saja dulu Vim dengan konfigurasi *default*:

```
$ vim contoh.txt
```

Anda sebenarnya juga dapat meng-*install* **gVim**, versi *graphical* dari Vim dengan fitur *menu* pada bagian atas *window* sehingga dapat diakses dengan perangkat *mouse*. Namun sekali lagi, tutorial ini mengasumsikan kita tidak menggunakan bantuan perangkat *mouse* sama sekali, sehingga aplikasi Vim pun sudah cukup.

Setelah menjalankan perintah di atas, Anda akan mendapatkan tampilan kurang lebih sebagai berikut:



Mulai dari sini Anda mungkin mulai bingung, bagaimana cara kita mengetik? Perlu diketahui bahwa Vim mengenal 3 *mode* yang harus kita mengerti dulu, yakni *Insert Mode*, *Normal Mode* dan *Insert Mode*. Secara *default*, Vim berada

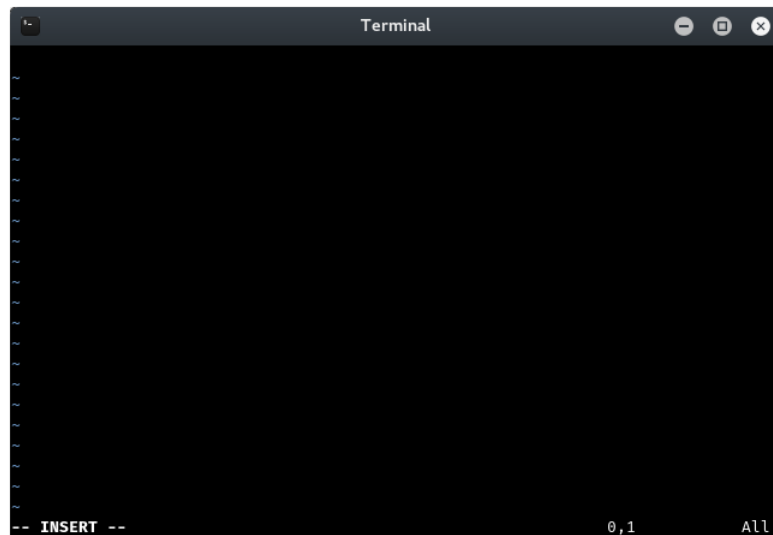
³Penulis menggunakan sistem operasi Kali Linux yang berbasis Debian dan aplikasi `gnome-terminal` sebagai *terminal emulator*

pada *Command Mode*, dan Anda diharapkan selalu kembali ke *mode* ini setiap Anda selesai melakukan penyuntingan teks.

Mari kita pahami dulu arti dari masing-masing *mode* tersebut:

- *Normal Mode* adalah *mode* di mana Anda dapat menjalankan perintah (*command*). *Mode* ini merupakan *mode* ketika Anda menjalankan Vim pertama kali.
- *Insert Mode* adalah *mode* di mana Anda memasukkan teks.
- *Visual Mode* adalah *mode* di mana Anda dapat menyorot secara *visual* sekumpulan teks, sehingga Anda dapat melakukan operasi penyuntingan pada teks tersebut.

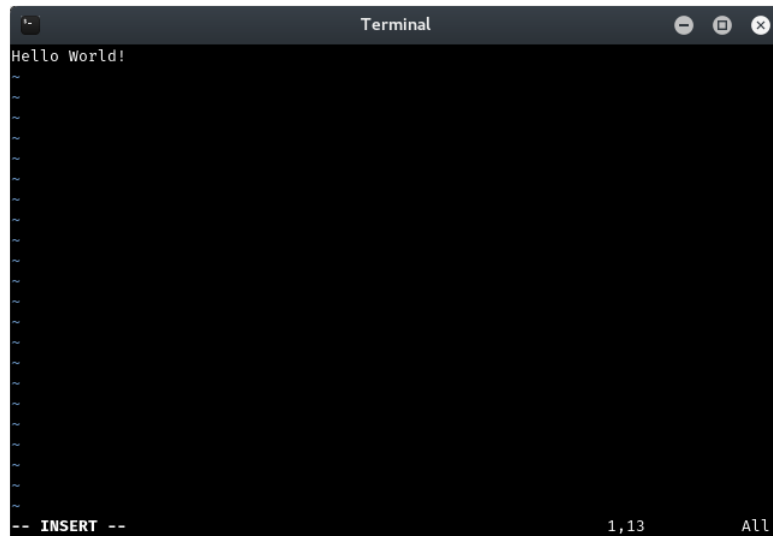
Untuk memulai proses pengetikan, tekan tombol `i`, dan kalau Anda perhatikan bagian bawah layar Vim Anda akan berubah menjadi `-- INSERT --`



Sekarang Anda dapat mulai mengetik, misal `Hello World!`

Setiap selesai mengetik, Anda disarankan untuk kembali ke *command mode*, hal ini dapat dilakukan dengan menekan tombol `Esc`⁴. Perhatikan sekarang tanda `-- INSERT MODE --` sekali lagi menghilang, dan ini berarti Anda berada pada *command mode*.

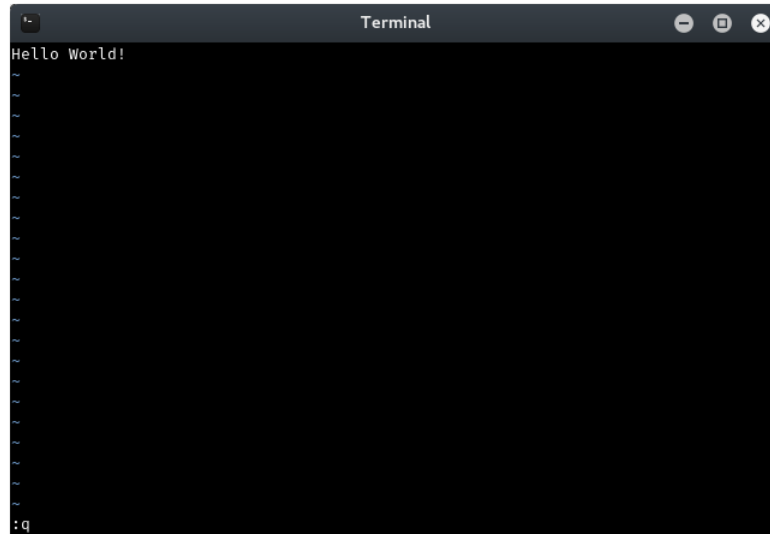
⁴Selain `Esc`, Anda dapat juga menggunakan kombinasi `Ctrl+c` atau `Ctrl+[` atau Anda dapat mengkonfigurasi pilihan tombol sesuai selera Anda dengan membuat konfigurasi berkas `.vimrc` yang akan dibahas pada bagian lebih lanjut



Sekarang simpan berkas ini dengan mengetik `:w`, sehingga muncul tampilan seperti berikut:



Masih dalam *command mode*, sekarang tekan `:q` untuk keluar (*exit*) dari Vim.



Mari kita rangkum, perintah-perintah yang sudah Anda lakukan di atas:

```
i    =>  beralih ke insert mode
Esc =>  beralih ke command mode
:w   =>  menyimpan berkas
:q   =>  keluar dari Vim
```

Selamat! Anda baru saja membuat berkas `contoh.txt` dan mengisikan kalimat `Hello World!` ke dalam berkas tersebut! Memang, pada awalnya terasa rumit, namun ketika Anda sudah terbiasa dan mulai menguasai beberapa kombinasi tombol perintah lebih lanjut, lambat laun Anda akan merasakan apa yang dimaksud dengan “*The power of Vim*”.

3 Navigasi

Seperti disebutkan di atas bahwa penggunaan Vim yang efisien adalah dengan meniadakan penggunaan perangkat *mouse* dan mengoptimalkan penggunaan perangkat *keyboard*. Hal ini juga berarti bahwa Anda diharapkan mengetahui bagaimana mengetik dengan *keyboard* secara efisien.

Salah satu teori yang sering dipakai adalah *Home Row Technique*, yakni menempatkan keempat jari kiri pada tombol A, S, D, F dan keempat jari kanan pada tombol J, K, L, ;. Dengan teknik ini, Anda dapat menjangkau tombol lain, baik yang berada di atas maupun di bawah tombol-tombol di atas secara lebih baik.

Hebatnya Vim juga menggunakan teori tersebut secara intensif, daripada menggunakan tombol panah yang biasanya terletak pada sebelah kanan *keyboard* dan sulit dijangkau, Vim menggunakan tombol H, J, K, L untuk navigasi ke kiri, ke bawah, ke atas, dan ke kanan.

```

h => Bergerak ke kiri satu karakter
j => Bergerak ke bawah satu baris
k => Bergerak ke atas satu baris
l => Bergerak ke kanan satu karakter

```

Memang pada awalnya perlu waktu untuk penyesuaian, namun ketika sudah mahir, lihatlah bagaimana kecepatan mengetik Anda dapat meningkat secara signifikan karena Anda tidak perlu ‘menggeser’ tangan kanan Anda untuk menjangkau tombol panah.

Anda dapat mengkombinasikan tombol navigasi di atas dengan angka untuk melakukan navigasi secara lebih cepat, misal:

```

10h => ke kiri 10 karakter
15j => ke bawah 15 baris
5k => ke atas 5 baris
4l => ke kanan 4 karakter

```

dan beberapa kombinasi tombol lainnya:

```

Ctrl+F => menuju ke bawah satu layar
Ctrl+B => menuju ke atas satu layar
Ctrl+D => menuju ke bawah setengah layar
Ctrl+U => menuju ke atas setengah layar
M      => menuju ke bagian tengah dari layar
H      => menuju ke bagian paling atas dari layar
L      => menuju ke bagian paling bawah dari layar

```

Bila sudah mahir, Anda dapat mencoba melakukan navigasi berdasar kata (*word*), misal:

```

w => menuju ke kata berikutnya
W => menuju ke kata berikutnya (berdasar spasi)
b => menuju ke kata sebelumnya
B => menuju ke kata sebelumnya (berdasar spasi)
e => menuju ke akhir kata
E => menuju ke akhir kata (berdasar spasi)
ge => menuju ke akhir kata sebelumnya
gE => menuju ke akhir kata sebelumnya (berdasar spasi)
8w => menuju ke kata kedelapan dari posisi kursor sekarang
5b => menuju ke kiri 5 kata dari posisi kata sekarang
3e => menuju ke akhir karakter dari kata ketiga

```

Pertanyaannya apa beda *w* dengan *W* (atau *b* dengan *B*)?

Navigasi dengan *w* masih memperhitungkan tanda *punctuation* seperti , . : / ! dll, sedangkan *W* hanya memperhitungkan spasi. Ambil contoh kalimat berikut:

```

http://www.infotiket.com menyediakan layanan info tiket
pesawat, konser, kereta api dan hotel

```

Ketik perintah berikut:

```

^ => berpindah ke awal baris
w => berpindah ke :
W => kursor bergerak ke huruf m pada kata 'menyediakan'

```


Anda dapat juga melakukan navigasi menggunakan tombol **f** untuk pencarian ke kanan dan **F** untuk pencarian ke kiri. Sebagai contoh kalimat berikut:

Saya sedang belajar Vim

Kursor Anda sekarang berada pada awal kalimat, dan Anda ingin menuju ke kata Vim, Anda dapat melakukannya dengan kombinasi tombol **4w** atau Anda dapat juga dengan menggunakan perintah:

```
fV
```

Tombol di atas berarti cari ke arah kanan karakter **V**. Sekarang kursor Anda dengan cepat berpindah ke huruf **V**. Perlu diingat tombol pencarian ini bersifat *case sensitif*, sehingga pada contoh di atas karena yang kita cari adalah huruf **V** kapital, maka kita harus menekan tombol pada **keyboard f** diikuti dengan **Shift+v**.

Untuk pencarian ke kiri, gunakan huruf **F** (f kapital), diikuti dengan huruf yang ingin Anda cari. Anda dapat mengulangi pencarian berikutnya dengan menekan tombol **;**. Misal pada kalimat berikut:

Tutorial Belajar Vim Versi 0.0.8 (Vladimir)

Untuk menuju ke kata **Vladimir**, gunakan kombinasi tombol berikut:

```
fV;;
```

dan kursor Anda pun dengan cepat berpindah ke huruf **V** pada kata **Vladimir**. Jika Anda ingin kembali ke huruf **V** sebelumnya, tekan tombol koma (,).

```
^fV;;;,
```

Perintah di atas mencari huruf **V** pada kata **Vladimir**, kemudian kembali ke huruf **V** pada kata **Vim**.

Perlu diingat bahwa tombol **f** ini hanya dapat digunakan untuk mencari karakter pada satu baris saja, dan sifatnya lebih kepada navigasi daripada fungsi pencarian. Untuk fungsi pencarian kata pada dokumen, akan dibahas pada bagian lain.

Selain fitur **f**, Vim juga memiliki fitur tombol **t** yang berfungsi seperti **f**, bedanya **t** akan bergerak ke 1 karakter sebelum karakter yang dicari. Untuk lebih jelasnya perhatikan contoh berikut:

Anda memiliki kalimat:

Vim merupakan program penyunting teks yang hebat.

Anda ingin mencari 1 karakter sebelum huruf **a** pada kata **hebat** (yakni huruf **b**). Berikut kombinasi tombol yang dapat Anda lakukan.

```
^
ta;;;;
```

Selanjutnya navigasi untuk menuju ke awal atau akhir baris, dapat digunakan tombol:

```
0 => menuju ke awal baris
^ => menuju ke karakter pertama dalam sebuah baris
$ => menuju ke akhir baris
```

Menuju ke baris tertentu secara cepat dapat menggunakan perintah:⁵

```
:n      => menuju ke baris ke-n
:100    => menuju ke baris ke-100
```

Anda juga dapat melakukan perintah di atas dengan:

```
10G     => menuju ke baris 10
100G    => menuju ke baris 100
```

Menuju ke awal dokumen atau akhir dokumen:

```
gg => menuju ke awal dokumen (baris pertama)
G  => menuju ke akhir dokumen (baris terakhir)
```

Selain kombinasi tombol di atas, kita juga dapat menggerakkan layar tanpa merubah posisi kursor, sehingga posisi kursor tepat berada di tengah layar, dengan menekan tombol:

```
zz => reposisi layar sehingga kursor tepat di tengah layar
```

4 Fungsi Penyuntingan

4.1 Menyisipkan Teks

Berikut beberapa kombinasi tombol perintah untuk melakukan penyuntingan teks secara cepat pada Vim:

```
i  => merubah ke insert mode pada posisi kursor
I  => menuju ke kursor kosong pada awal baris dan merubah ke insert mode
a  => bergerak ke kanan 1 karakter dan merubah ke insert mode
A  => menuju ke kursor kosong pada akhir baris dan merubah ke insert mode
o  => membuat baris baru di bawah posisi kursor sekarang dan merubah ke insert mode
O  => membuat baris baru di atas posisi kursor sekarang
dan merubah ke insert mode
s  => hapus huruf pada posisi kursor dan beralih ke insert mode
S  => hapus baris pada posisi kursor dan beralih ke insert mode
```

Selain kombinasi tombol di atas, Anda juga dapat menggunakan kombinasi perintah `ct` diikuti dengan karakter 'sampai dengan' yang Anda tuju, untuk lebih jelasnya perhatikan contoh berikut:

Semisal Anda memiliki kata `getUrl` yang ingin Anda menjadi `findUrl`, maka Anda dapat menggunakan perintah berikut (pastikan posisi kursor berada pada huruf `g` pada kata `getUrl`):

```
ctU     => change til U
find    => ubah menjadi find
Esc     => beralih ke normal mode kembali
```

⁵Navigasi menggunakan nomor baris lebih saya sukai daripada harus menekan tombol M, H, L karena lebih akurat dan praktis, tapi itu semua tergantung selera Anda..

4.2 Salin dan Tempel

Kedua fungsi ini saya yakin banyak dipakai ketika kita sedang bekerja dengan teks, dan Anda mungkin terbiasa menggunakan perangkat *mouse* untuk melakukan kedua fungsi tersebut.

Sekali lagi, *mouse is your enemy!*, jadi mari kita lakukan fungsi tersebut dengan *keyboard way*. Anda dapat menyalin sebuah baris dengan mudahnya menggunakan perintah `yy` atau `Y`, kemudian menuju ke baris yang Anda inginkan, kemudian tekan `p` untuk menempelkan pada baris di bawah kursor, atau tekan `P` untuk menempelkan pada baris di atas kursor.

Anda dapat juga menempelkan baris tersebut sebanyak yang Anda inginkan, misal Anda ingin menempelkan sebanyak 5 kali pada posisi di bawah kursor Anda sekarang, maka tekan tombol `5p`, maka otomatis baris yang Anda menyalin akan tersalin sebanyak 5 kali.

Contoh:

```
Vim is so powerful
```

Anda ingin menyalin kalimat di atas sebanyak 9 kali, maka perintah di Vim adalah:

```
Y
9p
```

Hasil:

```
Vim is so powerful
Vim is so powerful
Vim is so powerful
Vim is so powerful
Vim is so powerful
Vim is so powerful
Vim is so powerful
Vim is so powerful
Vim is so powerful
Vim is so powerful
```

Atau Anda ingin menyalin kata `so powerful` saja?

Tempatkan kursor pada huruf `s` pada kata `so`, kemudian ketik perintah berikut:

```
^      => ke awal baris
fs;    => cari s yang kedua
y2w    => salin 2 kata
o      => buat baris baru
Esc    => beralih ke normal (command) mode
p      => put (paste) in here
```

Hasilnya:

```
so powerful
```

Bagaimana? Sudah mulai merasakan *the power of Vim*??
Mari kita rangkum perintah yang ada dalam bagian ini:

```
Y => salin baris
yy => salin baris
yw => salin satu kata di sebelah kanan kursor
yb => salin satu kata di sebelah kiri kursor
y2w => salin dua kata di sebelah kanan kursor
p => put hasil salinan
```

4.3 Sorot / Visual Mode

Anda mungkin terbiasa melakukan fungsi sorot ini menggunakan perangkat *mouse*, dengan melakukan klik kiri, kemudian *drag* sampai daerah yang Anda inginkan. Vim juga memungkinkan melakukan itu, meski tentunya proses sorot menggunakan *keyboard*.

Vim memiliki 3 bentuk *visual* yang berbeda:

1. *per-character visual mode* (v)
2. *line visual mode* (V)
3. *block visual mode* (Ctrl+v)

Contoh *per-character visual mode*

Anda memiliki kalimat berikut:

```
Vim is great!
```

Anda dapat menyorot dan mengganti kata **great** menjadi **superb** dengan melakukan perintah berikut:

```
^      => menuju ke awal baris
2w     => menuju ke kata ke 3
v      => aktifkan fungsi sorot
e      => menuju ke akhir huruf pada kata
c      => menuju ke change mode
superb => ganti ke superb
```

Contoh *line visual mode*:

Asumsikan kita mempunyai baris teks sebagai berikut:

```
1 import os
2 from bottle import route, run
3
4 @route('/')
5 def homepage():
6     return 'Hello World!'
7
8 run(host='localhost', port=8080)
```

Katakanlah Anda ingin menyorot baris 4 sampai 6, dengan Vim Anda dapat melakukannya dengan

```
:4      => menuju ke baris ke 4
V       => sorot baris tersebut
2j      => sorot 2 baris dibawahnya
```

Jika Anda perhatikan ada perubahan warna pada daerah yang sedang Anda sorot, selanjutnya terserah Anda, apakah ingin disalin (menggunakan tombol y) atau dihapus (potong) (menggunakan tombol d).

Contoh *block visual mode*:

Dengan *visual block mode*, Anda dapat melakukan berbagai langkah manipulasi teks dengan lebih cepat. Perhatikan contoh berikut (dikombinasikan dengan perintah I (*insert*):

```
Ini contoh1.txt
Ini contoh2.txt
Ini contoh3.txt
Ini contoh4.txt
Ini contoh5.txt
```

Anda ingin menambahkan kata berkas pada setiap baris:

```
^       => menuju ke awal baris
w       => bergerak satu word
Ctrl+v  => mengaktifkan blok sorot
4j      => ke bawah 4 baris
I       => berubah ke mode insert
berkas  => ketik kata berkas
space   => memberi jarak 1 spasi dengan kata berikutnya
Esc     => kembali ke command mode
```

Dan hasilnya:

```
Ini berkas contoh1.txt
Ini berkas contoh2.txt
Ini berkas contoh3.txt
Ini berkas contoh4.txt
Ini berkas contoh5.txt
```

Amazing

Contoh lain

Misal Anda memiliki baris kalimat seperti berikut:

```
Ini baris yang panjang
Pendek
Ini baris yang panjang
```

Anda ingin menambahkan kata **sangat** antara kata **yang** dengan kata **panjang**.

```
^       => menuju ke awal baris
3w      => menuju ke kata ke-3
Ctrl+v  => aktifkan sorot blok
2j      => menuju 2 baris kebawah
I       => berubah ke insert mode
sangat  => ketik sangat
space   => memberi spasi antara kata
Esc     => kembali ke command mode
```

Dan hasilnya:

```
Ini baris yang sangat panjang
Pendek
Ini baris yang sangat panjang
```

Kita dapat lihat, dengan perintah `I`, baris kedua tidak berubah, karena memiliki panjang baris yang tidak sama.

Contoh berikutnya dikombinasikan dengan perintah `c` (*change*):

```
Ini baris yang sangat panjang
Pendek
Ini baris yang sangat panjang
```

Mari kita ubah kata `sangat` menjadi `--SANGAT--`

```
^          => menuju ke awal baris
3w         => menuju ke kata ke-3
Ctrl+v     => aktifkan blok sorot
2j         => menuju ke 2 baris dibawahnya
e          => menuju ke akhir kata sangat
c          => change
--SANGAT-- => ketik kata pengganti
Esc        => kembali ke command mode
```

Dan hasilnya:

```
Ini baris yang --SANGAT-- panjang
Pendek
Ini baris yang --SANGAT-- panjang
```

Contoh berikutnya merupakan kombinasi antara *visual block* dengan perintah `A` (*append*):

```
Ini baris yang panjang
Pendek
Ini baris yang panjang
```

Anda ingin menambahkan kata `sangat` di antara kata `yang` dengan kata `panjang`.

```
^          => menuju ke awal baris
3w         => menuju ke 3 kata ke kanan
h          => ke kiri 1 karakter
Ctrl+v     => aktifkan blok sorot
2j         => menuju ke 2 baris dibawahnya
A          => append mode
sangat     => ketik sangat
space      => beri jarak antar kata
Esc        => kembali ke command mode
```

Dan hasilnya, kata `sangat` ditambahkan pada setiap baris! Ini adalah perbedaan perintah `A` dengan perintah lainnya pada mode blok sorot.

```
Ini baris yang sangat panjang
Pendek          sangat
Ini baris yang sangat panjang
```

Contoh berikut ini mengkombinasikan blok sorot dengan perintah \$ dan A untuk menambahkan kata di setiap baris yang ada.

```
Ini baris yang panjang
Pendek
Ini baris yang panjang
```

Tambahkan kata **sekali** di setiap baris

```
$      => menuju ke akhir baris
Ctrl+v => aktifkan blok sorot
2j     => sorot 2 baris ke bawah
A      => append mode
Space  => beri jarak antar kata
sekali => ketik sekali
Esc    => kembali ke command mode
```

Dan hasilnya:

```
Ini baris yang panjang sekali
Pendek sekali
Ini baris yang panjang sekali
```

Contoh berikutnya merupakan kombinasi antara blok sorot dengan perintah **r** (*replace*).

```
Ini baris yang panjang
Pendek
Ini baris yang panjang
```

Mari kita ubah kata **panjang** menjadi kata **xxxxxx**

```
^      => menuju ke awal baris
3w     => ke kanan 3 kata
Ctrl+v => aktifkan blok sorot
2j     => sorot 2 baris ke bawah
e      => ke akhir kata
r      => replace mode
x      => ganti ke huruf x
```

Dan hasilnya:

```
Ini baris yang xxxxxx
Pendek
Ini baris yang xxxxxx
```

Contoh berikutnya menggeser bagian dari baris agar sejajar dengan baris di bawah (atas) nya.

```
Nama      : Sopier
Alamat    : Jogja
Pekerjaan : Wiraswasta
Pendidikan : Sarjana
```

Anda ingin menggeser 2 baris paling atas, supaya tanda : sejajar dengan baris dibawahnya:

```
^      => menuju ke awal baris
f:     => menuju ke tanda :
Ctrl+v => aktifkan blok sorot
j      => sorot 1 baris dibawahnya
>      => geser satu tab ke kanan
```

Dan hasilnya:

```
Nama       : Sopier
Alamat     : Jogja
Pekerjaan  : Wiraswasta
Pendidikan : Sarjana
```

Anda juga dapat menggeser ke kiri dengan mengganti tanda > menjadi <.

Tips:

Anda dapat melakukan sorot ulang teks yang sudah Anda sorot sebelumnya dengan menggunakan perintah `gv`.

4.4 Fungsi Hapus / Potong

Vim memiliki fungsi hapus yang sangat handal dan efisien, perhatikan kombinasi tombol berikut:

```
x  => menghapus 1 karakter pada posisi kursor
X  => menghapus 1 karakter di depan posisi kursor
d  => menghapus daerah yang sedang disorot
dd => menghapus satu baris
dw => menghapus satu kata ke depan
db => menghapus satu kata ke belakang
D  => menghapus dari posisi kursor sampai ke akhir baris
```

4.5 Undo dan Redo

Berikut ini tombol perintah untuk melakukan *undo* dan *redo* pada Vim:

```
u      => Undo
U      => Undo semua perubahan pada baris
Ctrl+R => Redo
:e!    => Membatalkan semua perubahan pada berkas
```

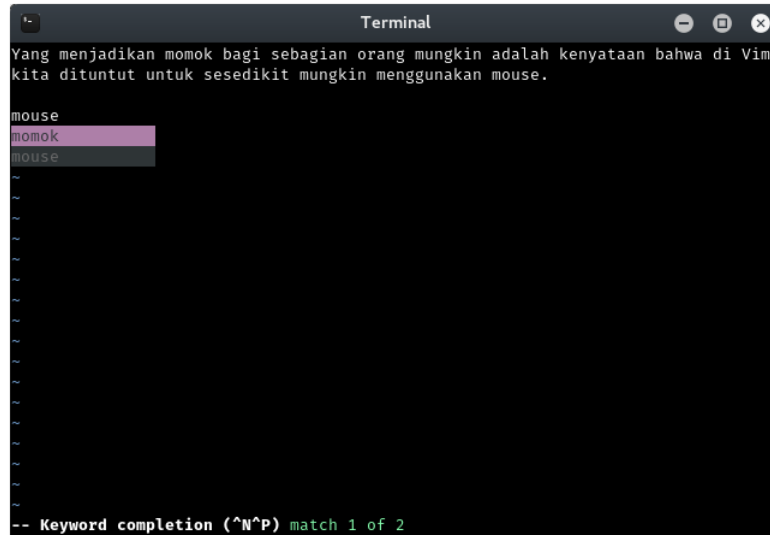
4.6 Word Completion

Sebagian orang mungkin menyebutnya *auto completion*, apa pun itu, cara kerjanya adalah dengan mengetikkan beberapa karakter kemudian Vim akan mencari dalam dokumen, kata yang cocok dengan sekumpulan karakter tersebut.

Berikut contohnya:

Yang menjadikan momok bagi sebagian orang mungkin adalah kenyataan bahwa di Vim kita dituntut untuk sesedikit mungkin menggunakan mouse.

Semisal Anda ingin mengetik ulang kata `mouse`, daripada harus mengetik secara utuh Anda cukup mengetik `mo` diikuti dengan tombol `Ctrl+p`, nanti Vim akan menunjukkan kata apa saja yang diawali dengan `mo`.



Anda dapat juga menggunakan kombinasi tombol `Ctrl+n` untuk pencarian ke bawah (*bottom*).

<code>Ctrl+p</code>	=> pencarian ke atas (up)
<code>Ctrl+n</code>	=> pencarian ke bawah (bottom)

4.7 Filename Completion

Sebagai seorang *developer*, Anda mungkin pernah dihadapkan pada sebuah kode di mana Anda harus menulis *filepath* ke dalam kode tersebut. Vim memberikan kemudahan untuk itu, tanpa kita harus mengingat-ingat di mana berkas tersebut berada.

Perhatikan contoh kode berikut:

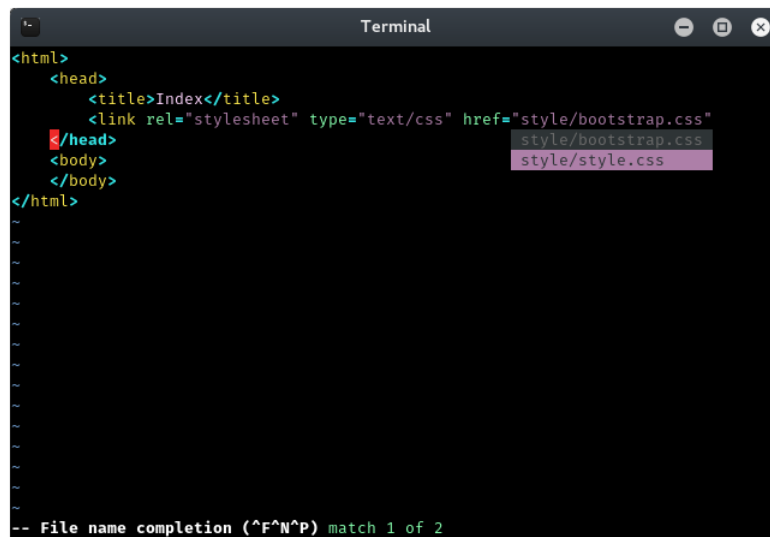
```
<html>
<head>
  <link rel="stylesheet" href="somefilepath.css"/>
</head>
<body>
</body>
</html>
```

Dengan Vim, Anda dapat menyunting `somefilepath.css` dengan cara berikut: Pastikan Anda berada pada baris yang akan disunting

```
^
f";;
ci"
```

[opsional] Ketik direktori di mana Anda mau mencari (misal /home/)
Ctrl+x Ctrl+f
Tekan Ctrl+f atau Ctrl+n untuk bergerak maju (forward)
Tekan Ctrl+p untuk bergerak mundur (backward)
Jika sudah, tekan Ctrl+x lagi

Berikut ini tampilan ketika kita sedang menggunakan fitur *filename completion* dalam Vim.



```
Terminal
<html>
  <head>
    <title>Index</title>
    <link rel="stylesheet" type="text/css" href="style/bootstrap.css"
  </head>
  <body>
  </body>
</html>
~
~
~
~
~
~
~
~
~
~
-- File name completion (^F^N^P) match 1 of 2
```

4.8 Replace Karakter

Biasanya, ketika kita ingin mengganti sebuah karakter, kita menghapus karakter lama, masuk ke *insert mode*, kemudian memasukkan karakter yang baru. Kita dapat mempersingkat langkah tersebut dengan menggunakan tombol **r** diikuti dengan karakter pengganti.

Contoh pada kata Vim, kita ingin mengganti V kapital menjadi v kecil, caranya tempatkan kursor pada V, tekan **r**, kemudian v, dan karakter pun berubah.

Anda dapat juga menggunakan tombol **R** untuk mengganti beberapa huruf sekaligus, perhatikan contoh berikut:

Mr. Joni sedang belajar Vim

Ganti xxxx menjadi Joni dengan mengetikkan perintah berikut:

```
^      =>   menuju ke awal baris
W      =>   menuju ke huruf x pertama
R      =>   beralih ke replace mode
Joni   =>   ketik Joni untuk mengganti xxxx
```

Another little improvement on speed typing :)

4.9 Menyunting Kata dalam tanda ' ' (

Anda mungkin pernah berhadapan dengan teks seperti berikut ini:

```
<title>Judul</title>
```

Anda ingin mengganti kata Judul?

```
dit    => menghapus kata di dalam html tags
i      => beralih ke insert mode
```

Atau Anda memiliki kalimat berbentuk kurang lebih seperti berikut?

```
<a href="http://www.infotiket.com">Info Tiket</a>
```

Anda ingin menyunting kata yang berada dalam tanda "? Tempatkan kursor pada baris tersebut dan lakukan langkah berikut:

```
di"    => hapus kata di dalam tanda "
ci"    => hapus kata di dalam tanda " dan beralih ke insert mode
```

Untuk penyuntingan kalimat / kata di dalam tanda lainnya, cukup ganti tanda " dengan tanda yang Anda inginkan, misal (di(, di[, di{, dst).

Selain menggunakan dit, ada juga perintah:

```
cit    => change inside tags
yit    => copy inside tags
vitp   => visual inside tags then put
```

Silakan coba, Anda pasti menyukainya...

5 Fungsi Pencarian

Saya yakin fungsi ini banyak dipakai ketika kita sedang menyunting berkas. Bagaimana menggunakan fitur pencarian ini pada Vim?

5.1 Pencarian Case Insensitive

Vim pada dasarnya dapat melakukan pencarian pada satu berkas, banyak berkas atau pada daerah tertentu yang kita inginkan.

Perintah dasarnya adalah menggunakan tombol / diikuti dengan kata yang ingin kita cari, kemudian secara otomatis Vim akan mencari dan meng-*highlight* kata tersebut (jika ada).

Contoh:

```
/Vim    => mencari kata Vim forward
Enter   => enter
n       => next match
N       => previous match
```

Anda juga dapat menggunakan tombol ? untuk pencarian kebelakang (*backward*), berikut contohnya:

```
?Vim    => mencari kata Vim backward
Enter   => enter
n       => previous match
N       => next match
```

Anda juga dapat melakukan pencarian menggunakan tombol `*`, caranya tempatkan kursor Anda pada kata yang ingin Anda cari (misal `Vim`), kemudian tekan tombol `*`, maka secara otomatis Vim akan melakukan pencarian kata yang cocok dengan kata pada posisi kursor.

Tombol `#` memiliki fungsi yang sama dengan `*`, hanya saja modus pencarian untuk `#` adalah ke belakang (*backward*). Anda juga dapat menggunakan `g*` atau `g#` untuk mencari kata yang *non-exact*.

Sebagai contoh:

```
kata
katak
```

Anda ingin mencari semua kata yang mengandung kata `kata` di dalamnya? Lakukan perintah berikut:

Letakkan kursor pada kata `kata`, kemudian:

```
g*
```

Maka semua kata yang mengandung kata `kata` akan di *highlight* oleh Vim.

5.2 Pencarian Case Sensitive

Secara *default*, Vim menggunakan modus pencarian dengan pengaturan *case insensitive*, ini artinya kata `Vim` dengan `vim` dipandang sebagai satu kata yang sama. Pertanyaan selanjutnya bagaimana cara kita mencari dengan modus *case sensitive* dengan Vim?

Terlebih dahulu, Anda dapat mengaktifkan modus `smartcase` pada Vim dengan mengetikkan perintah berikut:⁶

```
:set smartcase
```

Kemudian lakukan perintah berikut:

```
/vim\C
atau
/\Cvim
```

Perintah di atas berarti mencari kata `vim`, bukannya `Vim`.

⁶Saya juga menambahkan baris tersebut pada berkas `.vimrc` pada direktori home

5.3 Fungsi Cari dan Ganti

Apabila Anda ingin melakukan fungsi cari dan ganti pada satu berkas utuh, Anda dapat menjalankan perintah berikut:

```
:%s/kata_asal/kata_ganti/g
```

Atau jika Anda ingin membatasi pencarian hanya pada baris tertentu:

```
:420, 421s/Anda/Kami/g
```

Perhatikan tanda spasi di belakang koma.

Atau Anda hanya ingin cari dan ganti pada satu baris saja?

```
:s/Anda/Kami/g
```

Atau mungkin Anda lebih suka sistem sorot daerah tertentu kemudian baru melakukan cari dan ganti pada daerah yang Anda sorot?

```
V
5j
:s/Anda/Kami/g
```

Perintah di atas berarti, sorot baris pada posisi kursor sampai 5 baris di bawah posisi kursor, kemudian cari kata **Anda** dan ganti dengan kata **Kami**.

Tanda **%** berarti melakukan pencarian pada seluruh baris di dokumen, jika Anda ingin melakukan fungsi cari dan ganti pada satu baris saja, maka hilangkan tanda **%**.

Tanda **g** berarti melakukan fungsi ini pada semua keterulangan (*occurences*) pada baris. Jika tidak menggunakan tanda **g**, maka Vim hanya akan mengganti kata pertama yang ditemukan pada baris.

Contoh:

```
aku dan kau bagaikan langit dan bumi
```

Kemudian ketikkan perintah berikut

```
V          => sorot baris
:s/dan/dengan/ => ganti kata dan yang pertama saja
```

Dan hasilnya:

```
aku dengan kau bagaikan langit dan bumi
```

Bandingkan jika kita menggunakan tanda **g**.

```
V          => sorot baris
:s/dan/dengan/g => ganti semua kata dan pada baris
```

Dan hasilnya:

```
aku dengan kau bagaikan langit dengan bumi
```

Anda juga dapat menentukan apakah fungsi ini bersifat *case sensitive* atau *case insensitive*, secara *default*, Vim menggunakan sifat *case insensitive*, jika Anda ingin melakukan secara *case sensitive*, Anda dapat menambahkan penanda I. Perhatikan perintah berikut:

```
:%s/anda/kami/gI
```

Perintah di atas hanya akan merubah kata **anda**, tapi tidak dengan kata **Anda**.

Dengan menggunakan penanda c, maka Anda akan dihadapkan pada konfirmasi interaktif, apakah Anda akan melakukan penggantian pada kata yang sudah ditemukan.

```
:%s/anda/kami/gcI
```

Perintah di atas berarti, cari di seluruh dokumen, di seluruh baris, kata **anda**, dan ganti menjadi **kami**, dengan sebelumnya menanyakan konfirmasi pada Anda, dengan format kurang lebih seperti ini:

```
replace with kami (y/n/a/q/l/^E/^Y)?
```

Selanjutnya Anda dapat menekan y untuk *yes*, n untuk *no*, dan seterusnya...

Berikut ini bentuk perintah untuk mencari kata secara tepat (*exact match*) pada Vim. Mari kita gunakan contoh berikut:

```
andai  
seandainya  
andaikan
```

Jika Anda menggunakan perintah

```
:%s/andai/jika/g
```

Maka hasilnya:

```
jika  
sejikanya  
jikakan
```

Dan bisa dibilang, hasilnya kacau. Kita dapat menggunakan pencarian dengan modus *exact match* untuk mengatasi hal ini.

Pencarian modus *exact match* menggunakan bentuk sebagai berikut

```
:%s/\<kata_yang_dicari\>/kata_ganti/g
```

Perhatikan kita menambahkan tanda \< dan \> pada awal dan akhir kata yang ingin kita cari. Sehingga perintah di atas kita ubah menjadi sebagai berikut:

```
:%s/\<andai\>/jika/g
```

Dan hasilnya:

jika
seandainya
andaikan

Anda juga dapat melakukan fungsi cari dan ganti dikombinasikan dengan pencarian pola dengan **regular expression**, namun materi tersebut tidak akan dibahas dalam tutorial ini, jika berminat silakan di-*explore* sendiri.

6 Macro

6.1 Dasar Macro

Fitur ini adalah fitur yang sangat-sangat saya sukai, karena sesuai dengan prinsip DRY (*Dont Repeat Yourself*). Ambil contoh teks html berikut ini:

```
<ul>
  satu
  dua
  tiga
  empat
  lima
  enam
  tujuh
  delapan
  sembilan
  sepuluh
</ul>
```

Semisal Anda ingin menambahkan tags `` dari satu sampai sepuluh, daripada bercepek menambahkan satu-persatu, Anda dapat membuat **macro** kemudian menjalankan **macro** tersebut sebanyak yang Anda inginkan.

Tempatkan kursor pada kata `satu`, kemudian jalankan kombinasi perintah berikut:

```
qa
I
<li>
Esc
A
</li>
Esc
j
q
9@a
```

Hasil:

```
<ul>
  <li>satu</li>
  <li>dua</li>
  <li>tiga</li>
  <li>empat</li>
  <li>lima</li>
```

```

        <li>enam</li>
        <li>tujuh</li>
        <li>delapan</li>
        <li>sembilan</li>
        <li>sepuluh</li>
    </ul>

```

6.2 Melakukan Penomoran Secara Otomatis

Ambil contoh berikut, saya ingin membuat 10 daftar kata python dengan nomor berurutan dari 1 sampai 10. Dengan bermodal satu baris berikut, saya dapat membuat 10 daftar kata python lengkap dengan nomor yang berurutan.

```
1. python
```

Berikut perintahnya di Vim:

```

qa
Y
p
Ctrl+A
q
8@a

```

Dan hasilnya adalah sebagai berikut:

```

1. python
2. python
3. python
4. python
5. python
6. python
7. python
8. python
9. python
10. python

```

So so so efficient, right?

7 Multi-tab

Seringkali Anda harus bekerja dengan banyak berkas sekaligus, di dunia IDE Anda mungkin sudah tidak asing lagi dengan fitur *multi-tab*, di mana Anda dapat membuka banyak berkas sekaligus dan berpindah antara satu berkas dengan berkas lain semudah melakukan klik pada berkas yang diinginkan.

Vim juga mengenal sistem *tabbing* seperti itu, berikut beberapa perintah ketika Anda bekerja dengan banyak *tab*:

<code>:tabnew</code>	=> membuat tab baru
<code>:tabnext</code>	=> berpindah ke tab berikutnya
<code>:tabprev</code>	=> berpindah ke tab sebelumnya
<code>:gt</code>	=> go to next tab
<code>:gT</code>	=> go to prev tab
<code>:tabfind</code>	=> mencari tab berdasar nama berkas
<code>:tabclose</code>	=> menutup tab

Ketika Anda mengaktifkan fitur *tab*, maka pada layar Vim bagian atas akan muncul *tab* baru selayaknya *tab* yang Anda lihat pada IDE lainnya, cuman disini warnanya hitam dan putih :)

Untuk lebih lengkapnya, Anda dapat mengetikkan perintah `:tab` diikuti dengan tombol `Tab` untuk melihat perintah-perintah apa saja terkait dengan fitur ini.

8 Registers

Untuk meningkatkan efisiensi dalam pekerjaan penyuntingan teks, Vim memiliki fitur *registers*, di mana Anda dapat menyimpan apa yang sudah Anda salin atau hapus ke dalam sebuah *key* tertentu.

Ketika sudah tersimpan, Anda dapat menambahkan apa yang sudah Anda simpan atau menyalinnya ke tempat yang Anda inginkan.

Bentuk *syntax* perintah *registers* pada Vim adalah sebagai berikut:

"kyy

Perintah di atas berarti salin sebuah baris (*y*) kemudian simpan baris tersebut ke dalam tombol *k*. Jika Anda ingin menampilkan isi dari *register* tersebut, Anda dapat melakukan perintah berikut:

"kp

Perintah tersebut berarti *put* atau taruh isi dari *register k* pada posisi kursor sekarang.

Anda dapat menambahkan isi sebuah *register* dengan menggunakan huruf kapital dari *register* yang Anda buat sebelumnya.

"Kyy

Perintah di atas berarti salin baris pada posisi kursor, kemudian tambahkan (*append*) baris tersebut ke dalam *register k*.

Perhatikan contoh berikut:

Vim adalah program penyunting teks yang handal.

Tekan "kyy untuk menyalin baris di atas.

Kemudian Anda memiliki sebuah baris baru lagi

Namun, proses belajar Vim memang tidak mudah

Berikutnya tekan "Kyy untuk menambahkan baris ini ke dalam *register k*.

Kemudian *put* isi dari *register k* ke dalam sebuah baris, dan hasilnya:

Vim adalah program penyunting teks yang handal.

Namun, proses belajar Vim memang tidak mudah

Selain menyimpan salinan, Anda dapat juga menyimpan hapusan ke dalam *register*, caranya tentu dengan mengganti perintah salin dengan perintah hapus. Perhatikan contoh perintah berikut:

```
"kdd => hapus dan simpan sebuah baris ke dalam register k
"Kdd => tambahkan hapusan berikutnya ke dalam register k
"kp  => taruh isi dari register k ke posisi kursor sekarang
```

9 Marks

Sesuai dengan artinya, *Marks* pada Vim berfungsi sebagai penanda posisi, sehingga Anda dapat dengan mudah menuju kembali ke posisi tersebut.

Marks pada Vim disimbolkan dengan huruf alfabet dari a–z untuk tiap berkas, dan huruf kapital A–Z untuk penanda global. Jika Anda sedang menyunting 10 berkas, tiap berkas dapat memiliki penanda posisi a, namun hanya memiliki 1 penanda posisi A.

Perintah untuk mengaktifkan penanda pada posisi kursor adalah dengan menekan tombol m diikuti dengan huruf sebagai penanda.

Misal:

```
ma => beri tanda pada posisi kursor sekarang dengan huruf a sebagai penanda
```

Untuk kembali pada posisi tersebut, tekan tanda petik tunggal (') atau tanda *backtick* (`).

```
'a => menuju ke awal baris di mana penanda berada
```

```
`a => menuju tepat ke posisi kursor di mana penanda berada
```

Selanjutnya Anda pun dapat menyalin, menghapus atau pun mengubah teks dengan penanda ini sebagai tujuan akhir. Misalnya:

Vim merupakan aplikasi penyunting teks yang hebat.

Jalankan perintah berikut:

```
^      => menuju ke awal baris
fp;;   => menuju ke huruf p pada kata penyunting
ma     => beri tanda a pada posisi ini
^      => kembali ke awal baris
d'a    => hapus dari posisi kursor sampai penanda a
```

Beberapa perintah lain dari fungsi *marks* pada Vim:

```
:marks      => daftar semua penanda yang aktif
:marks aB   => daftar penanda a, B
:delmarks a  => hapus penanda a
:delmarks a-d => hapus penanda a,b,c,d
:delmarks!   => hapus semua penanda huruf kecil
```

Fungsi navigasi dengan penanda:

```
] '      => menuju ke baris penanda berikutnya
[ '      => menuju ke baris penanda sebelumnya
] `      => menuju ke posisi kursor penanda berikutnya
[ `      => menuju ke posisi kursor penanda sebelumnya
```

Yang perlu diingat, gunakan ' (tanda petik tunggal) untuk menuju ke awal baris di mana penanda berada, atau gunakan ` (*backticks*) untuk menuju ke posisi kursor di mana penanda berada.⁷

⁷tanda backtick berada di sebelah kiri angka 1 pada keyboard

10 Buffer

Satu lagi fitur yang handal dari Vim untuk bekerja dengan banyak berkas adalah *buffer*. Saya sendiri lebih menyukai ini dibandingkan dengan sistem *tabbing*, karena layar kita tetap bersih, seolah-olah bekerja dengan satu berkas, padahal sebenarnya banyak berkas yang sedang kita sunting.

Berikut beberapa perintah terkait dengan *buffer*:

<code>:badd</code>	=>	menambahkan berkas / buffer baru
<code>:ls</code>	=>	melihat berkas-berkas yang sedang kita sunting
<code>:bd</code>	=>	menghapus buffer (bukan menghapus berkas)
<code>:b <angka></code>	=>	berpindah ke buffer <angka> sesuai dengan urutan pada perintah <code>ls</code>
<code>:b <nama berkas></code>	=>	berpindah ke buffer berdasar nama
<code>:bn</code>	=>	berpindah ke next buffer
<code>:bp</code>	=>	berpindah ke prev buffer

Berikut ini contoh perintah `:ls`:

```
:ls
1 %a + "vim_docs.tex"           line 593
2 #h  "~/vimrc"                 line 1
3 h   "[No Name]"               line 0
```

Begini cara baca keluaran dari perintah `:ls` di atas:

<code>%</code>	=>	buffer aktif yang sedang dilihat
<code>#</code>	=>	alternate buffer, tekan <code>Ctrl+~</code> untuk berpindah ke alternate buffer
<code>h</code>	=>	hidden buffer (tidak sedang dilihat)
<code>+</code>	=>	ada perubahan dan belum disimpan
<code>-</code>	=>	inactive buffer
<code>line xxx</code>	=>	menunjukkan di baris berapa kursor Anda berada

Terlihat di sana saya sedang menyunting 3 berkas, di mana berkas yang aktif saya sunting saat ini adalah berkas nomor 1 (ditandai dengan `%`).

Kalau saya ingin pindah ke berkas `.vimrc`, saya tinggal perintahkan `:b 2` atau `:b vimrc` atau `:bn`.

Selain menghapus satu per satu, Anda juga dapat menghapus banyak *buffer* sekaligus dengan menggunakan *range*, contoh:

```
:1,5bd    => menghapus buffer 1 sampai 5
```

Jika Anda ingin menghapus sebuah *buffer* tanpa melakukan penyimpanan, Anda dapat menambahkan tanda perintah `!` seperti pada contoh berikut:

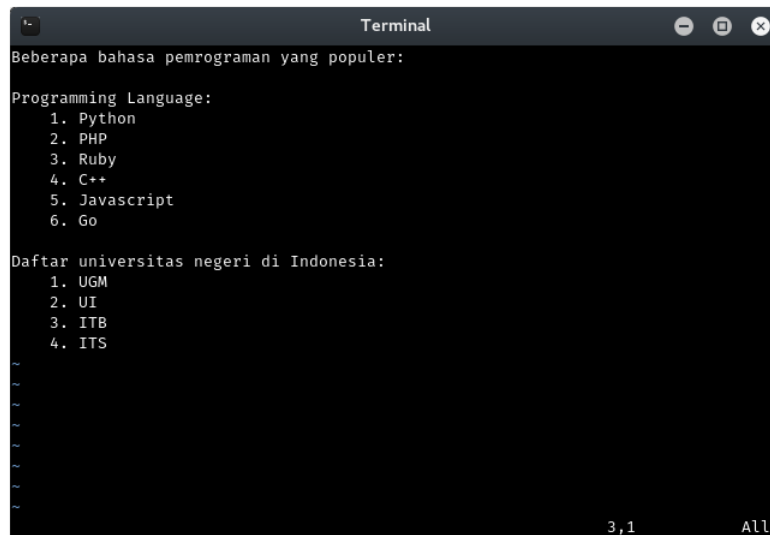
```
:bd!
```

Anda dapat juga melakukan pencarian ke dalam banyak *buffer* sekaligus, atau mungkin cari dan ganti ke banyak *buffer* sekaligus. Selanjutnya, silakan di-*explore* sendiri kemampuan dari *buffer* ini.

11 Code Folding

Bagi Anda yang menulis ribuan baris teks atau pun kode, fitur ini pasti sangat berguna buat Anda untuk membuat tulisan atau kode Anda “lebih enak” dilihat.

Berikut ini tampilan kode sebelum dan sesudah menggunakan *folding* di Vim:



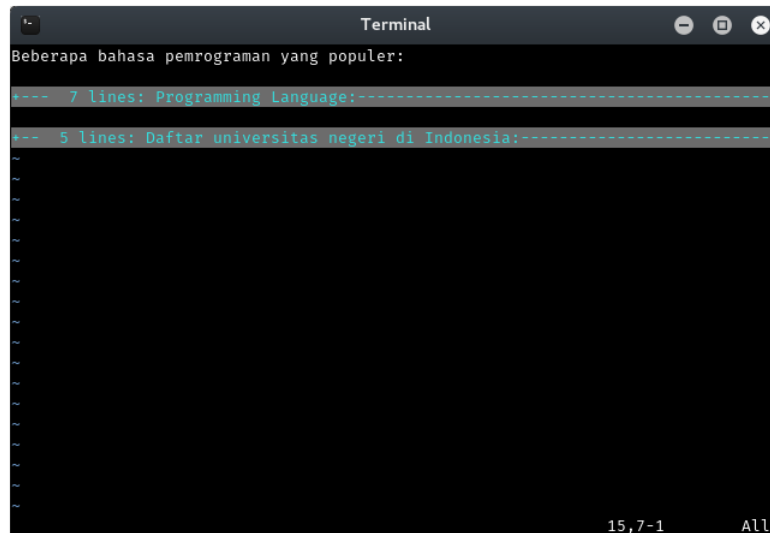
A terminal window titled "Terminal" showing a list of programming languages and Indonesian universities. The text is as follows:

```
Beberapa bahasa pemrograman yang populer:

Programming Language:
1. Python
2. PHP
3. Ruby
4. C++
5. Javascript
6. Go

Daftar universitas negeri di Indonesia:
1. UGM
2. UI
3. ITB
4. ITS
~
~
~
~
~
```

The status bar at the bottom right shows "3,1" and "All".



The same terminal window after applying code folding. The sections for programming languages and universities are now collapsed into single lines with fold markers. The text is as follows:

```
Beberapa bahasa pemrograman yang populer:

+--- 7 lines: Programming Language:-----
+--- 5 lines: Daftar universitas negeri di Indonesia:-----
~
~
~
~
~
```

The status bar at the bottom right shows "15,7-1" and "All".

Vim memiliki 6 cara dalam melakukan *folding*:

1. `manual`
2. `indent`
3. `syntax`
4. `marker`
5. `expr`
6. `diff`

Untuk sementara ini mari kita pelajari 3 dari 6 cara di atas, yakni cara `manual`, `marker` dan `indent`.

Folding manual dapat Anda lakukan dengan terlebih dahulu menyorot baris yang ingin Anda lipat, kemudian lipat (buka lipatan) dengan menekan:

```
zf => melipat baris
za => membuka lipatan
```

Contoh:

```
Jadual hari ini:
1. Rapiin kamar kerja
2. Cuci motor
3. Mandi
4. Sarapan
5. Coding
6. Makan siang
7. Ngopi2 sama temen
```

Posisikan kursor pada jadual pertama, kemudian tekan:

```
V => sorot baris pertama
6j => sorot 6 baris dibawahnya
zf => lipat baris
```

Atau cara lebih singkat:

```
zf6j
```

Selain dengan teknik sorot, Anda pun dapat menggunakan teknik `marks` yang dapat Anda pelajari di bagian lain tutorial ini.

Menggunakan contoh di atas, kita dapat memanfaatkan fitur `marks` untuk melakukan pelipatan:

Posisikan kursor pada jadual nomer 7, kemudian tekan:

```
ma => beri tanda dan simpan ke register a
6k => menuju ke jadual nomer 1
zf'a => lipat baris sampai mark a
```

Berikutnya pelipatan dengan metode `marker`, sebelumnya ketik perintah berikut ini:

```
:set foldmethod=marker
```

Selanjutnya Anda dapat memberikan tanda khusus pada bagian-bagian yang ingin Anda lipat. Perhatikan kode `LaTeX` berikut:

```
\section{Pendahuluan}
%{{{
Tulisan-tulisan ini sebenarnya bukanlah tutorial lengkap
yang mengajarkan kepada Anda seluk-beluk program Vim,
melainkan sekedar catatan pribadi penulis yang coba
dituangkan kedalam sebuah berkas elektronik dengan tujuan
untuk dokumentasi pribadi, syukur-syukur kalau ada pihak
lain yang membacanya dan mampu mendapatkan manfaat dari
tulisan-tulisan ini.
%}}}
```

Kemudian tambahkan penanda (sebagai contoh `%{{{ %}}}`):

```
\section{Pendahuluan}
%{{{
Tulisan-tulisan ini sebenarnya bukanlah tutorial lengkap
yang mengajarkan kepada Anda seluk-beluk program Vim,
melainkan sekedar catatan pribadi penulis yang coba
dituangkan kedalam sebuah berkas elektronik dengan tujuan
untuk dokumentasi pribadi, syukur-syukur kalau ada pihak
lain yang membacanya dan mampu mendapatkan manfaat dari
tulisan-tulisan ini.
%}}}
```

Secara otomatis, Vim akan melihat tanda tersebut, letakkan kursor di dalam penanda tersebut, kemudian tekan:

```
za => melipat kode
za => lakukan lagi untuk membuka lipatan
```

Atau Anda pun dapat melipat semua bagian yang sudah Anda beri penanda dengan mengetikkan:

```
zM      => melipat semua bagian yang sudah diberi marker
```

dan untuk membukanya:

```
zR      => membuka semua lipatan yang sudah diberi marker
```

Anda dapat menentukan sendiri **marker** yang akan dipakai, dengan memberikan perintah:

```
:set foldmarker=/*,*/
```

Perintah di atas, berarti **marker** pembuka ditandai dengan tanda `/*` dan **marker** penutup dengan tanda `*/`.

Metode berikutnya adalah dengan metode **indent**. Teknik ini berguna ketika Anda memiliki kode yang memiliki indentasi terstruktur (misal bahasa **Python**).

Sebelumnya ketik perintah berikut:

```
:set foldmethod=indent
```

Perhatikan kode **Python** berikut:

Secara *default*, Vim memiliki kemampuan tersebut, berikut ini kombinasi tombol perintah yang Anda perlukan untuk mengakses kemampuan jejalah berkas pada Vim:

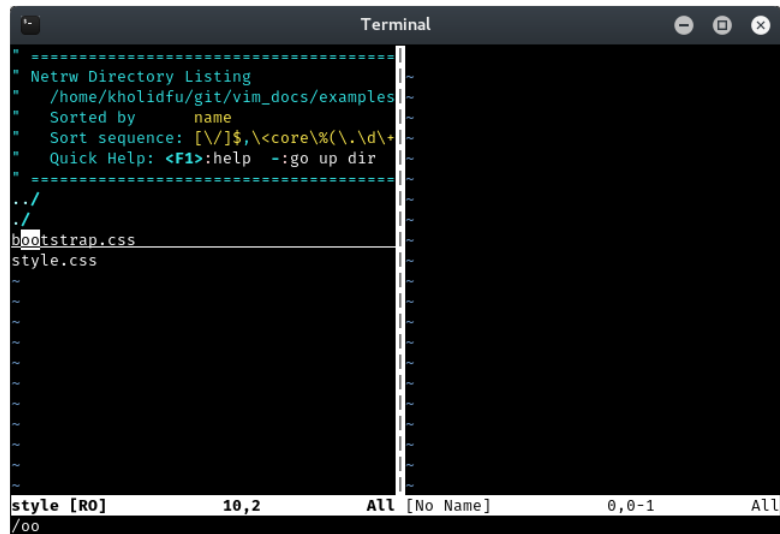
```
:edit .    => membuka file explorer
:e.        => membuka file explorer
:sp.       => membuka file explorer dengan horizontal split
:vs.       => membuka file explorer dengan vertical split
```



Ketika Anda berada pada fitur jejalah berkas, Anda dapat menggunakan kombinasi tombol berikut untuk membuat berkas baru, direktori baru, mengganti nama, dan menghapus berkas:

```
%  => membuat berkas baru
d  => membuat direktori baru
R  => mengubah nama berkas / direktori pada kursor
D  => menghapus berkas / direktori pada kursor
```

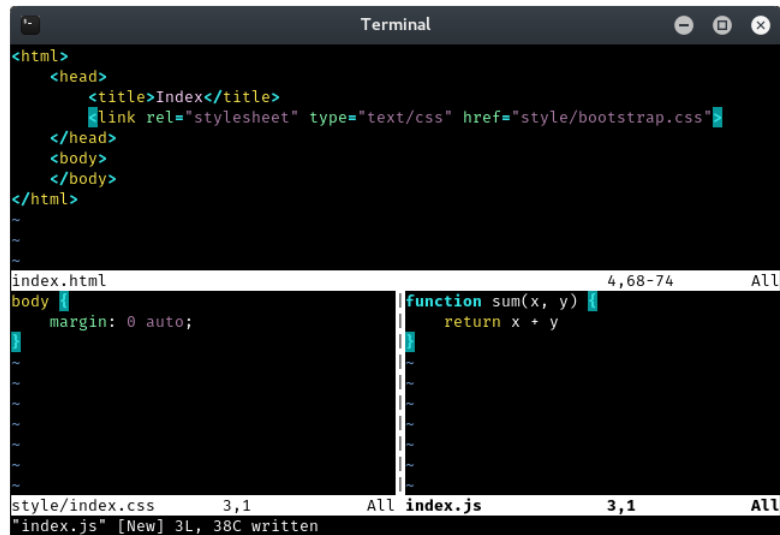
Karena fitur *file explorer* tidak lain adalah *buffer*, maka Anda dapat dengan mudah menutupnya dengan perintah `:bd` atau melakukan pencarian menggunakan `/` (*forward*), atau `?` (*backward*). Dan tentunya juga tombol `h j k l` untuk melakukan navigasi ke kiri, bawah, atas dan kanan.



13 Split Screen

Dalam bekerja dengan banyak berkas, Vim juga memiliki kemampuan untuk memecah layar menjadi beberapa bagian, baik itu horisontal maupun vertikal.

Perhatikan gambar berikut:



Terlihat saya sedang menyunting 3 berkas dan ketiga-tiganya terbuka, fi-

tur ini sangat membantu ketika kita bekerja dengan banyak berkas sekaligus. Bagaimana caranya:

```
:sp      => split horisontal
:vsp     => split vertical
Ctrl+w+w => berpindah antar window
Ctrl+w+r => berpindah antar window clockwise
Ctrl+w+R => berpindah antar window counter-clockwise
Ctrl+w+l => berpindah ke window sebelah kanan
Ctrl+w+h => berpindah ke window sebelah kiri
Ctrl+w+j => berpindah ke window sebelah bawah
Ctrl+w+k => berpindah ke window sebelah atas
Ctrl+w+- => memperkecil ukuran window (mode horizontal)
Ctrl+w++ => memperbesar ukuran window (mode horizontal)
:q       => menutup window
```

Anda dapat juga menggunakan perintah `:sp` atau `:vsp` diikuti dengan nama berkas yang ingin Anda sunting dalam *window* baru.

14 Session

Fungsi fitur ini adalah untuk menyimpan berkas-berkas yang Anda kerjakan sebelumnya, daripada membuka ulang satu-persatu berkas tersebut, Anda tinggal menyimpannya ke dalam *session* untuk kemudian dibuka kembali, dan otomatis Vim akan membuka berkas-berkas tersebut dalam *buffer*-nya.

Berikut cara kita mengelola *session* dalam Vim:

```
Menyimpan session
:mksession work1.vim

Memanggil session dari dalam Vim
:source work1.vim

Memanggil session dari terminal
$ vim -S work1.vim
```

15 Konfigurasi Vim

Anda dapat mengatur program Vim dengan membuat berkas `.vimrc` pada direktori *home* Anda.

```
$ vim ~/.vimrc
```

Pengaturan ini sendiri mungkin berbeda antara pengguna satu dengan pengguna lainnya, tergantung selera dan kebiasaan, berikut konfigurasi yang saya pakai:

```
set nocompatible

filetype on
filetype plugin on
```

```

filetype indent on
syntax on

set autowrite
set ruler
set hidden
set autochdir

colorscheme delek

set tabstop=8
set shiftwidth=4
set softtabstop=4
set expandtab

set showcmd
set number
set smartindent
set autoindent
set laststatus=2
set linespace=3

set wrap
set linebreak
set nolist
set incsearch
set hlsearch
set ignorecase
set smartcase

set foldenable
set mousehide
"set splitbelow

nmap <space> :

set wildmode=list:longest

imap jj <esc>

map <f2> :w\!!python %

```

Konfigurasi ini secara garis besar adalah standar, kecuali saya merubah tombol `:` menjadi `Space`, dan tombol `Esc` menjadi `jj`, semua ini dilakukan biar posisi tangan tidak bergeser ke kanan dan ke kiri.

Silakan dicoba, atau Anda mungkin memiliki preferensi sendiri, Vim memberikan kebebasan untuk itu..

16 Lain-lain

Berikut ini kumpulan tips dan trik yang sering saya pakai dan siapa tahu bermanfaat juga buat Anda...

16.1 Menjalankan Perintah Shell dari Vim

Ini termasuk salah satu fitur yang sangat saya sukai, Anda tidak perlu bolak-balik keluar dari Vim untuk sekedar menjalankan perintah **Shell**.

Sebagai contoh, dokumen ini ditulis menggunakan **LaTeX**, dan tentunya saya sering melakukan **compiling** dari format **.tex** ke **.pdf** untuk melihat apakah ada kesalahan penulisan atau tidak. Proses **compiling** itu sendiri menggunakan **shell command**:

```
$ pdflatex vimdocs.tex
```

Daripada harus keluar masuk Vim, saya dapat mengeksekusi perintah tersebut dari dalam Vim menggunakan perintah berikut:

```
:!pdflatex vimdocs.tex
```

Dan jika ingin mengulangi perintah terakhir, asya dapat dengan mudah mengetik berikut di Vim:

```
:!!
```

Sangat efisien bukan?

16.2 Menyisipkan Keluaran dari Shell Command ke dalam Vim

Semisal Anda ingin menyisipkan tanggal dan jam saat ini ke dalam dokumen Vim yang sedang Anda tulis, daripada harus capek-capek mengetik ulang, kita dapat menggunakan fungsi **read** berikut:

```
:r !date
```

Atau ingin menyisipkan kalender menggunakan perintah **shell cal**?

```
:r !cal
```

Maka, secara otomatis keluaran dari perintah **date** akan disisipkan pada baris di mana kursor Anda berada.

```
December 2012
Su Mo Tu We Th Fr Sa
                1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

16.3 Singkatan

Anda memiliki kata yang sangat sering Anda ulang-ulang dan ingin membuat versi singkatannya supaya lebih efisien dalam proses pengetikan? Vim memiliki fitur tersebut dengan perintah berikut:

```
:ab yg yang
```

Maka, secara otomatis setiap kali Anda mengetik `yg` diikuti dengan `spasi` atau `,`, maka kata tersebut berubah menjadi `yang`.

Secara lengkap, berikut perintah-perintah terkait dengan fitur *abbreviations* ini.

```
:ab dg dengan => dg diganti dengan
:ab            => daftar semua singkatan
:una dg       => hapus dg dari daftar singkatan
:abc         => hapus semua daftar singkatan
```

16.4 Membatasi Panjang Baris Maksimum n Karakter

Baris yang terlalu panjang kadang merepotkan, karena kita harus melakukan *scrolling* ke kanan untuk dapat membaca baris bagian kanan ujung. Untuk mengatasi hal tersebut, saya sendiri lebih suka mengatur Vim dengan panjang maksimum adalah 80 karakter:

```
:set textwidth=80
```

Konfigurasi ini otomatis memerintahkan Vim untuk memotong baris setiap kali kursor berada pada posisi ke-80.

Biar tidak berulang-ulang mengetikkan perintah tersebut, kita dapat memasukkan konfigurasi tersebut kedalam berkas `.vimrc`:

```
set textwidth=80
```

Selain cara otomatis tersebut, kita juga bisa memotong secara *manual* dengan mengetikkan perintah:

```
gq1
```

16.5 Menggabung Dua atau Lebih Baris

Lawan dari memotong baris yang panjang, adakalanya kita ingin menggabungkan beberapa baris menjadi 1 baris saja. Hal ini dapat kita lakukan dengan menempatkan kursor di baris paling atas dari beberapa baris yang ingin Anda gabung, kemudian ketikkan:

```
J      => jika hanya dua baris
JJ     => 3 baris
JJJ    => 4 baris
```

Ulangi menekan J sampai semua baris tergabung menjadi satu. Perintah ini secara otomatis akan menambahkan spasi diantara baris yang digabungkan. Jika Anda ingin menggabungkan baris tanpa spasi, gunakan perintah gJ.

Contoh:

```
Ini baris pertama
Ini baris kedua
Ini baris ketiga
Ini baris keempat
```

Tekan perintah berikut:

```
^      => menuju ke awal baris
J      => gabung garis 1 dan 2
J      => gabung lagi
gJ     => gabung lagi, kali ini tanpa spasi
```

Sehingga hasilnya sebagai berikut:

```
Ini baris pertama Ini baris kedua Ini baris ketigaIni baris keempat
```

16.6 Lowercase, Uppercase dan Titlecase

Bagaimana cara mengubah dari *lowercase* menjadi *uppercase* dalam Vim?

Perhatikan contoh kalimat berikut:

vim merupakan program penyunting teks yang hebat.

Anda dapat merubah kalimat di atas menjadi *uppercase* dengan menekan tombol berikut:

```
V      => sorot baris
U      => ubah menjadi uppercase, atau
u      => ubah menjadi lowercase
```

Berikut hasil ketika sudah dirubah menjadi *uppercase*

VIM MERUPAKAN PROGRAM PENYUNTING TEKS YANG HEBAT.

Untuk mengubah kata atau baris menjadi *titlecase*, kita dapat menggunakan fitur *search n replace* digabungkan dengan *regular expression*:

```
V      => sorot baris
:s/\w*/\u&/g  => ubah setiap awal kata menjadi huruf kapital
```

Dan hasilnya:

Vim Merupakan Program Penyunting Teks Yang Hebat.

Anda juga dapat melakukannya dengan cara *semi-manual*, berikut caranya:

```
^      => menuju ke awal baris
vU     => sorot huruf pada awal kata dan ubah menjadi uppercase
w.     => menuju ke awal kata berikutnya dan ulangi langkah sebelumnya
w.     => menuju ke awal kata berikutnya dan ulangi langkah sebelumnya
ulangi lagi sebanyak yang Anda inginkan
```

Atau Anda juga dapat memanfaatkan *plugin* bernama *titlecase* yang dapat Anda unduh di <http://www.vim.org>

Selamat mencoba ...

16.7 Cari dan Hapus Baris Berdasar Pola

Saatnya kita mencoba mempelajari fitur *regular expression* pada Vim. Pada bahasan kali ini kita akan mencari dan menghapus baris jika dalam tersebut terdapat pola (*pattern*) yang cocok dengan yang kita tentukan.

Perhatikan contoh berikut ini:

```
this is a line
these are also lines
those are line
that is another line
```

Kita dapat menghapus baris yang memiliki kata `lines` dengan mengetikkan perintah berikut:

```
:g/lines/d
```

atau menghapus kata yang mengandung `is`

```
:g/is/d
```

atau kata yang diawali dengan `th`

```
:g/^th/d
```

atau kata yang diakhiri dengan `ine`

```
:g/ine$/d
```

Anda dapat juga menentukan pada baris berapa sampai berapa fungsi ini diterapkan, contoh berikut akan menghapus baris yang memiliki kata `lines` pada baris 1296 sampai 1299:

```
:1296,1299g/lines/d
```

16.8 Membuat Baris Baru Identik

Berikut ini adalah perintah untuk membuat banyak baris yang identik:

```
5aVim is great!    =>  buat 5 baris baru dengan isi Vim is great!
Tekan enter        =>  buat baris baru
Tekan Esc          =>  kembali ke command mode
```

Dan hasilnya:

```
Vim is great!
Vim is great!
Vim is great!
Vim is great!
Vim is great!
```

16.9 Cara Lain Beralih ke Command Mode

Di bagian atas sudah disebutkan bahwa untuk beralih dari *insert mode* ke *command mode*, kita dapat menekan tombol **Esc** pada *keyboard*. Kenyataannya, bagi sebagian orang (termasuk saya), tombol **Esc** terlalu jauh letaknya dari *home row*.

Untuk itu sebagian pengguna Vim ada yang melakukan *remapping* tombol **Esc** ke tombol lain, misalnya saya *remap* tombol **Esc** ke tombol **jj**. Caranya adalah dengan menambahkan baris berikut pada berkas `.vimrc` Anda:

```
imap jj <esc>
```

Selain itu, sebenarnya Anda dapat juga menggunakan tombol **Ctrl+[** atau **Ctrl+c** untuk berpindah dari *insert mode* ke *command mode*.

Silakan pilih yang Anda suka, selama itu untuk meningkatkan produktivitas Anda dalam pekerjaan penyuntingan teks.

16.10 Bracket Matching

Tanda kurung biasanya selalu ditulis dalam bentuk berpasangan, ada (maka ada), begitu juga dengan tanda {} dan [].

Kalau Anda menulis kode program, biasanya pasangan dari tanda tersebut berada jauh di bawah atau kanan, Vim dalam hal ini memberikan kemudahan untuk mencari pasangan dari tanda-tanda tersebut. Dengan menekan tanda %, maka secara otomatis kursor akan bergerak ke pasangan tanda tersebut.

Lihat kode berikut:

```
require('casper').create({
  loadImages: false,
  verbose: true,
  logLevel: debug
});
```

Posisikan kursor Anda pada kurung buka " (" setelah kata **create**, kemudian tekan %, apa yang terjadi? Vim akan mencocokkan di mana letak kurung tutup yang merupakan pasangan dari kurung buka tadi, tekan % sekali lagi, dan Anda berpindah ke tempat kursor Anda sebelumnya.

Apa jadinya jika tidak ada pasangan kurung yang tepat? Vim tidak akan memberitahukan pesan kesalahan apa-apa, namun karena posisi kursor tidak berpindah, maka dapat disimpulkan bahwa kurung tersebut belum ditutup.

```
require('casper').create({
  loadImages: false,
  verbose: true,
  logLevel: debug
});
```


16.11 Mengaktifkan Penomoran Baris

Secara *default*, Vim tidak menampilkan penomoran baris, namun Anda dapat dengan mudah menampilkannya dengan mengetikkan:

```
:set number    => nomor baris aktif
:set nonumber  => nomor baris tidak aktif
```

16.12 Mengulang Perintah Terakhir

Tahukah Anda kalau Anda dapat mengulang perintah Anda yang terakhir kali menggunakan tanda titik (*dot*)?

Misal Anda memiliki teks sebagai berikut:

```
tahukah anda? vim sudah berumur 20 tahun lebih?
```

Anda ingin mengganti kalimat tersebut menjadi *titlecase*, bagaimana caranya?

```
^ => menuju ke awal baris yang berisi huruf
v => sorot huruf pertama pada kata pertama
U => ubah menjadi huruf kapital
w => bergerak ke kata berikutnya
. => ulangi perintah vU
```

Dan hasilnya...

```
Tahukah Anda? Vim Sudah Berumur 20 Tahun Lebih?
```

Anda dapat juga mengulang perintah terakhir yang Anda berikan ke Vim dengan mengetikkan:

```
@:
```

Sebagai contoh, Anda ingin membuka *tab* baru pada Vim dengan menggunakan perintah `:tabnew`. Sebentar kemudian Anda butuh untuk membuka *tab* baru satu lagi, daripada harus mengetik `:tabnew` lagi, Anda dapat mengulang perintah tersebut dengan cara `@:`, dan perintah yang paling terakhir pun, dijalankan ulang oleh Vim.

```
:tabnew    => membuka tab baru
@:         => buka tab baru lagi
:tabclose  => tutup tab
@:         => ulangi perintah tutup tab
```

16.13 Eksekusi Kode Bash dalam Vim

Selain dapat menjalankan kode Python, Vim juga dapat menjalankan perintah Bash, sebagai contoh Anda memiliki baris kode seperti berikut ini:

```
echo "hai from Bash"
date
echo "bye..."
```

Kemudian jalankan perintah berikut:

```
V2j    => sorot baris pada kursor dan 2 baris dibawahnya
:!bash => eksekusi kode
```

dan hasilnya:

```
hai from Bash
Fri Dec 21 06:30:10 WIT 2012
bye...
```

16.14 Mengetahui Nama Berkas yang sedang disunting

Daripada harus mengetik ulang nama berkas, Vim memberikan kemudahan dengan menggunakan **register %**, yang mana fungsinya adalah sebagai *shortcut* untuk berkas yang sedang disunting.

Jika Anda ingin menampilkan nama berkas yang sedang disunting, pada *insert mode*, ketik:

```
Ctrl+r
%
```

Atau, pada *command mode*:

```
"%p
```

dan hasilnya:

```
vim_docs.tex
```

Dan jika Anda ingin menampilkan nama berkas secara utuh dengan *path*-nya, pada *command mode*, ketik perintah berikut:

```
:put =expand('%:~')
```

Dan hasilnya:

```
/home/banteng/Dropbox/dokumentasi/vim/vim_docs.tex
```

16.15 Resize Splits

Sebagaimana layaknya sebuah *window* pada aplikasi *GUI* yang dapat diubah ukurannya dengan melakukan *click-and-drag*, Vim juga memiliki fitur tersebut ketika kita bekerja dengan *split screen*.

Perintah untuk mengubah ukuran layar horisontal berbeda dengan perintah untuk layar vertikal. Berikut ini perintah untuk mengubah ukuran layar pada mode *split* di Vim:

Mode *horizontal split*

Ctrl+w + => Increase
Ctrl+w - => Decrease
Ctrl+w _ => Maximize
Ctrl+w = => Equal size
10Ctrl+w + => Increase 10 lines
10Ctrl+w - => Decrease 10 lines

Mode *vertical split*

Ctrl+w > => Increase
Ctrl+w < => Decrease
Ctrl+w | => Maximize
Ctrl+w = => Equal size
10Ctrl+w > => Increase 10 chars
10Ctrl+w < => Decrease 10 chars