ویرایش ساده

ویم[1] یکی از قدرتمندترین و پربازده‌ترین ویرایشگران متن است چون کاربر می‌تواند با صرف زمان کمتری ویرایش کند. رسیدن به این قدرت و تابعیت هزینه زیادی داشته است.

در این فصل دستورات ابتدایی را یاد می‌گیرید تا بتوانید ویرایش را آغاز کنید:

–حرکت به چهار جهت

–نوشتن و پاک کردن متن

–استفاده از راهنمای برنامه

–خروج از برنامه!

پس ازفراگیری مطالب بالا می‌توانید دستورات پیشرفته‌تری یاد گیرید.

قبل از شروع

قبل از شروع باید ویم را نصب کنید، برای این کار ضمیمه الف را ببینید.

اگر از لینوکس استفاده می‌کنید این دستور را وارد کنید:

```
$ touch ~/.vimrc
```

اگر این فایل را نداشته باشید شما از ویم به صورت سازگار با Vi استفاده می‌کنید و بسیاری از قابلیت‌های ویم را از دست می‌دهید.

اگر از ویم در ویندوز استفاده می‌کنید خودکار فایلی به نام vimrc_ ساخته شده.

اجرای ویم برای اولین بار

برای اجرای ویم وارد کنید:

```
$ gvim file.txt
```

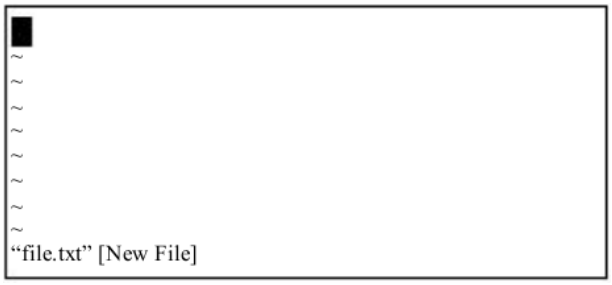
اگر از ویندوز استفاده می‌کنید:

```
C:> gvim file.txt
```

در این صورت ویم- فایل- file.txt-را باز می‌کند. چون این فایل جدید است شما یک پنجره‌ی خالی می‌بینید. خطوط مد '~' به معنای نبودشان در فایل اصلی است. به عبارت دیگر هر گاه تعداد خطوط فایل از خطوط موجود در صفحه نمایش کمتر باشد خطوط مد نمایش داده می‌شوند.

در پایین صفحه اعلانی هست که نشانگر نام فایل و جدید بودن آن است. این پیام موقتی است و به زودی جای خود را به اعلانی دیگر می‌دهد:

1 Vim



دستور gvim

دستور gvim یک پنجره جدید برای ویرایش ایجاد می‌کند. اگر از دستور vim استفاده کنید ویرایش درون پنجره فرمان انجام می‌شود. به عبارت دیگر با وارد کردن دستور vim در هر ترمینالی ویرایش درون همان ترمینال انجام می‌شود.

وضعیت‌ها

ویم یک ویرایشگر چند وضعی است یعنی در هر وضعی که باشد به طور خاصی عمل می‌کند و به فرمان‌های شما واکنش خاصی خواهد داشت.

اگر در پایین صفحه (جایی که نام فایل را دیدید) خالی باشد شما در وضع معمولی[2] هستید. در صورتی که در وضعیت درج[3] باشد آنجا عبارت --INSERT-- و در حالت دیداری[4] عبارت --VISUAL-- دیده می‌شود.

اولین ویرایش

در اینجا می‌آموزید فایلی را ویرایش کنید. در اینجا شما یاد نمی‌گیرید چگونه سریع و مفید ویرایش کنید، ولی یاد می‌گیرید ویرایش کنید!

وارد کردن متن

برای این که ویرایش کنید باید در وضعیت درج باشید. برای این کار کلید i را بزنید و به گوشه پایین نگاه کنید که --INSERT-- پدیدار شد (یعنی شما در حالت درج هستید.)

حالا هر چه می‌خواهید وارد کنید مهم نیست اشتباه بنویسید شما می‌توانید آن را تصحیح کنید. مثلا این شعر ناموزون برنامه نویسان را وارد کنید:

A very intelligent turtle

Found programming UNIX a hurdle

The system, you see,

Ran as slow as did he,

And that's not saying much for the turtle.

حالا کلید <ESC> را بزنید تا از حالت درج خارج شوید. (واو --INSERT-- غیب شد!)

صفحه شما چیزی مثل این شد:

2 Normal mode
3 Insert mode
4 Visual mode

```
A very intelligent turtle
Found programming UNIX a hurdle
        The system, you see,
        Ran as slow as did he,
And that's not saying much for the turtle.
~
~
~
~
```

خروج از بحران

ممکن است در مورد وضعیت[5] فعلی ویم سردرگم شوید. برای این که از هر وضعیتی که هستید خارج شوید و به حالت عادی بروید کلید <ESC> را بزنید.

حرکت در متن

در حالت معمولی می‌توانید در متن حرکت کنید. H چپ, J پایین, K بالا و L راست.

شاید فکر کنید این کلیدها شانسی انتخاب شده‌اند. کجا L به معنی راست است؟

ولی دلیل خوبی برای این کار هست. حرکت در متن بیشترین کاری است که در یک ویرایشگر انجام می‌شود. این کلیدها هم در ردیف اصلی صفحه کلید و زیر دست راست شما قرار دارند به عبارت دیگر جایی که شما می‌توانید سریعتر از همه جا فشارشان دهید.

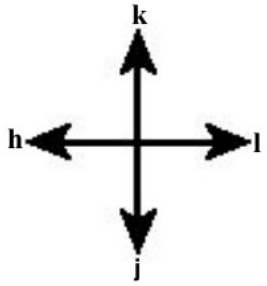
نکته

شما همچنان می‌توانید از کلیدهای جهتی صفحه کلید استفاده کنید. در این صورت سرعت کار خیلی کم می‌شود. چون شما مجبورید دستتان را از قسمت حروف بردارید و به قسمت کلیدهای جهتی ببرید. فکر کنید در یک ساعت صدها بار این کار را انجام دهید. پس از کلیدهای hjkl استفاده کنید.

یک راه برای این که مکان این کلیدها را به خاطر بسپرید چنین است:

h در چپ است و شما را به چپ می‌برد l هم سمت راست است و شما را به راست می‌برد j دسته‌ای به پایین دارد و k دسته‌اش به بالا است.

راه خوب دیگر هم این است که تصویری درست کنید و کنار صفحه نمایشتان بچسبانید:



حذف حرف

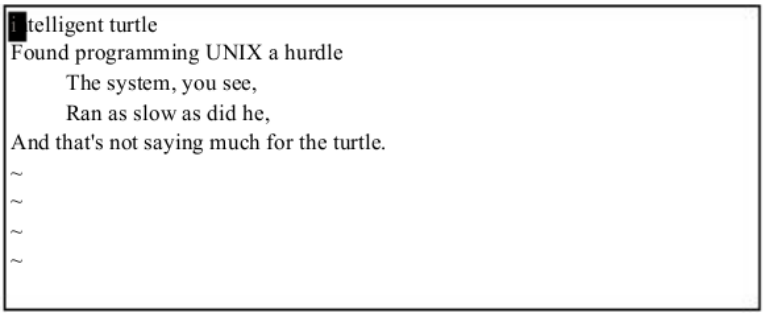
5 mode

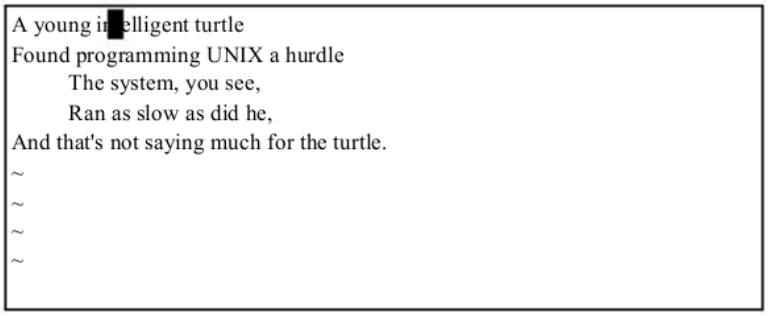
برای حذف یک حرف روی آن بروید و کلید x را بزنید.

به ابتدای خط اول بروید و بزنید xxxxxxxx (هشت x) سپس وارد کنید:

iA young <Esc>

با i نوشتن شروع می‌شود و با <ESC> از وضعیت درج خارج می‌شوید.





نکته

در ویم مثل دیگر ویرایشگرها باید برای رفتن به خط بعد از <ENTER> استفاده کنید. در غیر این صورت شما یک خط بسیار بلند خوهید داشت.

Undo و Redo

اگر شما بیشتر پاک کردید می‌توانید با u به وضع قبل برگردید. مثلا بروید و A young را پاک کنید تا خط اول چنین شود:

intelligent turtle

حالا کلید u را بزنید تاآخرین حرفی که پاک کردید برگردد:

g intelligent turtle

u بعد حرف قبلی را بر می‌گرداند:

ng intelligent turtle

و u بعدی ...:

ung intelligent turtle

oung intelligent turtle

young intelligent turtle

young intelligent turtle

A young intelligent turtle

اگر زیادی undo کردید می‌توانید redo کنید. CTRL+r یعنی undo خود را undo کنید!

نوع دیگر از u هم هست U . Uکل خط را به , وضعیت قبل از آخرین ویرایش بر می‌گرداند.

U بعدی, U قبلی که را لغو می‌کند.(اگر ویرایشی نکرده باشید)

نکته

undo چند سطحی در Vi کمی با ویم فرق دارد

نکته

من فرض کردم شما سازگاری با Vi را خاموش کرده‌اید(با ساختن .vimrc)

A very intelligent turtle

با very, xxxx را پاک کنید:

A intelligent turtle

turtle, xxxxxx را پاک کنید:

A intelligent

با U خط را به شکل اولیه بر می‌گردانید:

A very intelligent turtle

اگر دوباره کلید U را بزنید به شکل قبل بر می‌گردید:

A intelligent

خروج

دستور ZZ فایل را ذخیره و می‌بندد.

برخلاف بیشتر ویرایشگرها ویم از فایل شما پشتیبان نمی‌گیرد و شما دیگر راهی برای برگرداندن فایلی که از دست رفته ندارید. البته شما می‌توانید این قابلیت را فعال کنید.

رها کردن

شاید شما تغییراتی در فایل ایجادکرده باشید که ناگهان بفهمید اگر تغییرش نمی‌دادید بهتر بود و بهتر است فایل را بدون ذخیره نمودن ببنید.

نگران نباشید فقط وارد کنید:

:q!

اگر دوست دارید بیشتر بدانید، این دستور از سه قسمت تشکیل شده:

: که شما را وارد وضعیت دستور می‌کند

q به ادیتور می‌گوید بسته شو

! override command modifier است(!!!) ! لازم است چون ویم دوست ندارد چیزی را دور بریزد

چون این یک دستور بود برای اجرا باید کلید <ENTER> را بزنید(برای اجرای هر دستوری باید کلید <ENTER> را بزنید)

اگر برای خروج از q: استفاده کنید ویم به شما اخطار می‌دهد:

No write since last change (use ! to override)

با استفاده از ! به ویم می‌گویید که من می‌دونم این کار احمقانست ولی من بزرگ شدم و می‌خوام انجامش بدم!

چند دستور دیگر

شما چند دستور ساده یاد گرفتید حالا چند دستور پیچیده تر هم ببینید!

اضافه کردن عبارتی به انتهای خط

با دستور i نوشتن از قبل کرسر آغاز می‌شود. شاید شما بخواهید چیزی به آخر خط اضافه کنید. برای این کار از a[6] استفاده کنید.

مثلا می‌خواهید:

and that's not saying much for the turtle.

را به:

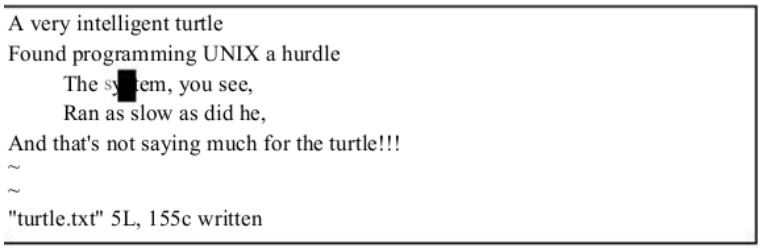
and that's not saying much for the turtle!!!

تبدیل کنید. برای این کار روی نقطه بروید و آن را با x پاک کنید. حالا کرسر روی e می‌رود. حالا وارد کنید:

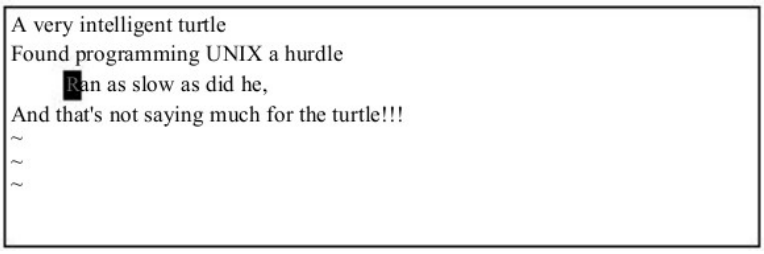
a!!!<ESC>

حذف یک خط

برای این کار وارد کنید dd. با این دستور کل خطی که کرسر در آن است پاک می‌شود (اهمیتی ندارد کرسر کجای خط است)



قبل



بعد

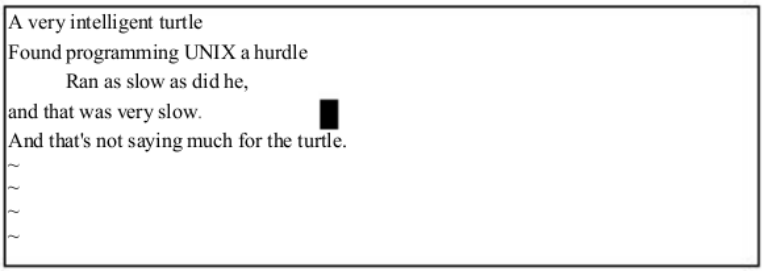
6 append

اضافه کردن یک خط

دستور o یک خط زیر خط فعلی کرسر اضافه می‌کند, کرسر را به ابتدا آن می‌برد و شما را در وضعیت درج قرار می‌دهد.

مثلا شما می‌خواهید زیر خط Ran چیزی اضافه کنید، بزنید o متن را وارد کنید و با <ESC> ازوضعیت درج خارج شوید.

اگر می‌خواهید خط جدید بالای کرسر باز شود از O استفاده کنید (بزرگ)



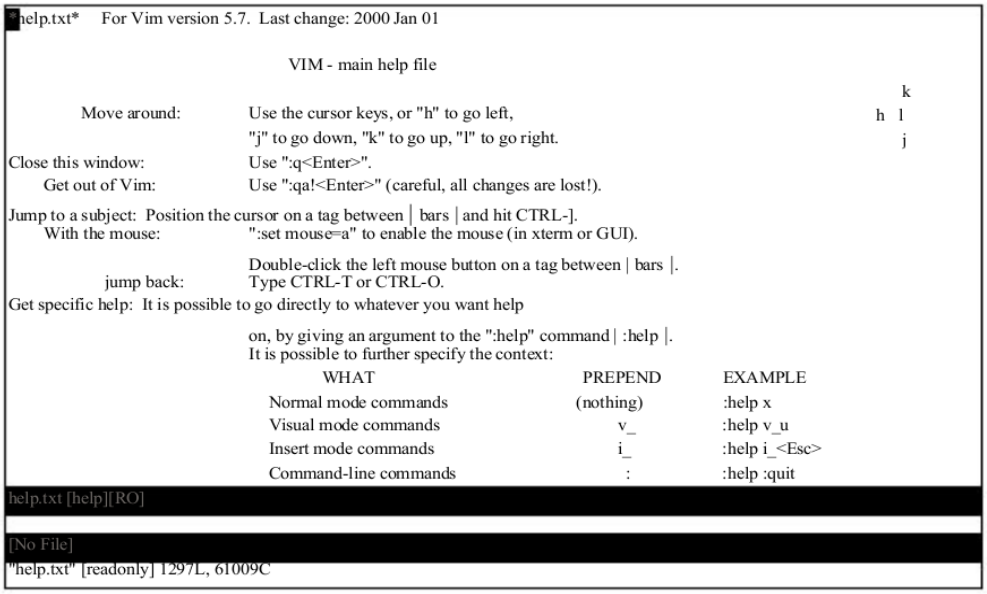
راهنما

و اینک مهمترین دستور! برای اینکه راهنما را ببینید بزنید:

:help

(یادتان باشد بعد از هر دستور باید <ENTER> کنید)

چیزی مثل این می‌بینید:



اگر دستور help را بدون موضوع بزنید فهرست کلی دستورات نشان داده می‌شوند.

شما با راهنما می‌توانید مثل یک فایل در حال ویرایش رفتار کنید و با HJKL در آن حرکت کنید.

وقتی شروع به خواندن راهنما می‌کنید می‌بینید بعضی از عبارات با بقیه فرق دارند. آن‌ها هایپرلینک هستند. کرسر را روی یکی از حروف آنها ببرید و بزنید [+Ctrl تا شما به آن برچسب بپرید و راهنمای موضوع مورد نظرتان را نشان دهد. برای برگشتن به برچسب قبلی از CTRL+T استفاده کنید.

بالای صفحه عبارت *help.txt* دیده می‌شود. ویم برای حرکت در برچسب‌ها از این عبارات استفاده می‌کند

برای این که راهنمایی موضوع خاصی را ببینید بزنید:

:help subject

مثلا دستور x:

:help x

یا اینکه راهنمایی برای پاک کردن:

:help deleting

شاید هم دوست دارید تمام دستورهای ویم را ببینید:

:help index

اگر هم می‌خواهید راهنمای یک دستور کنترلی را ببیند (مثلا CTRL+A):

:help CTRL-A

ویم معمولا راهنمای اجرای دستورات در حالت عادی می‌دهد. یعنی:

:help CTRL-H

راهنمای اجرای این دستور را در حالت عادی می‌دهد اگر بخواهید راهنمای اجرایش را در حالت درج ببینید بزنید:

:help i_CTRL-H

(جدول را ببینید):

نمونه	پیشوند	وضعیت
:help x	ندارد	حالت عادی
:help CTRL-u	CTRL-	دستورات کنترلی
:help v_u	V	حالت دیداری
:help i_<esc>	I	حالت درج
:help :q	:	حالت فرمان[7]
:help c_<DEL>	C	ویرایش خط فرمانی
:help -t	-	آرگومان‌های ورودی ویم
:help 'textwidth'	'	تنظیمات

برای نشان دادن کلیدهای خاص از علامت کوچکتر استفاده کنید مثلا برای کلید بالا:

:help <Up>

برای دیدن لیست کلیدهای خاص ضمیمه ب را نگاه کنید.

7 Ex mode

راه‌های دیگر ورود به راهنما

از کلید F1 استفاده کنید یا اگر صفحه کلید شما کلید help دارد آن را بزنید.

استفاده از شمارنده

اگر می‌خواهید ۹ خط بالا بروید به جای این که بزنید kkkkkkkkk بزنید 9k .

شما تمام حرکت‌های بعدی را فقط با یک عدد انجام دادید.

در این فصل جایی شما سه '!' به انتهای خط اضافه کردید می‌توانستید از شمارنده استفاده کنید:

3a!<Esc>

آموختار ویم

نسخه لینوکسی ویم با یک آموختار همراه است برای دیدن آن در خط فرمان وارد کنید:

$ vimtutor

شما می‌توانید آن‌چه این‌جا خواندید در درس یکم این آموختار ببینید.

در نسخه‌ها غیر لینوکسی از این دستور استفاده کنید:

:help tutor

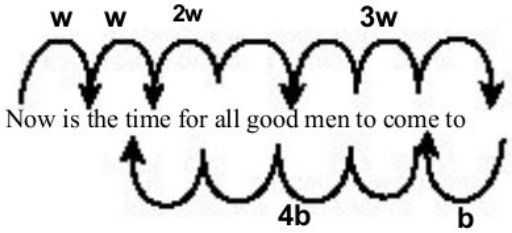
کمی سریعتر ویرایش کنید

دستورات ساده‌ای که در فصل اول معرفی شد به شما یاد می دهد ویرایش کنید.
در این فصل دستوراتی یاد می‌گیرید تا موثرتر ویرایش کنید. این دستورات این‌ها هستند.

–دستورات بیشتر برای حرکت در متن
–جستجوی سریع در یک فایل
–دستورات بیشتر برای پاک کردن و تغییر
–دستور جابجا کردن
–ماکرو (ذخیره و اجرای دستور ذخیره شده)
–کاراکترهای ویژه
در فصل از تعدد روش‌هایی که می‌توانید درمتن حرکت کنید شگفت‌زده می‌شوید.

Word Movement

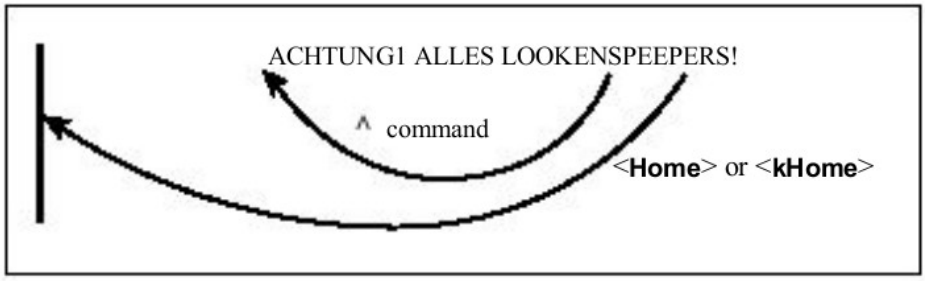
حرکت کلمه‌ای
برای اینکه کرسور به کلمه بعدی بپرد از دستور w استفاده کنید.
برای این که به کلمه قبلی بپرید از b استفاده کنید.



مثل بیشتر دستورات ویم شما می‌توانید برای یک دستور شمارنده بگذارید مثلا 4b به ۴ کلمه قبلی می‌پرد. تصویر را ببینید.

حرکت به اول یا آخر سطر

با دستور $ کرسر به انتهای سطر می‌رود.
البته شما می‌توانید از کلید <end> هم استفاده کنید.
این دستور هم می‌تواند با عدد همراه شود. مثلا دستور 1$ شما را به انتهای همان سطر می‌برد. 2$ به انتهای سطر بعد و ... .
دستور ^ شما را به اولین حرف غیر خالی (space یا tab نباشد) می‌برد.
کلید <home> شما را به ابتدای سطر می‌برد. عدد 0 هم همین کار را می‌کند.
(فرق <home> و ^ را در تصویر مشاهده کنید)
این دستورها هم می‌توانند با عدد همراه شود ولی با عدد کاری ندارد! (عدد شما تاثیری ندارد)



جستجو در یک سطر

یکی از دستورات مفید دستور جستجوی یک حرف در سطر است. دستور fx شما را به اولین x بعد از کرسر می‌برد.
مثلا شما در ابتدای چنین سطری هستید.

To err is human. To really foul up you need a computer.

شما می‌خواهید به h از human بروید. خوب بزنید fh

To err is human. To really foul up you need a computer.

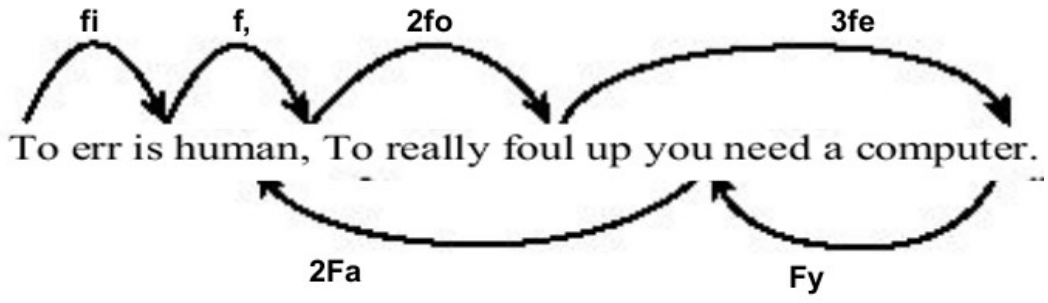
برای رفتن به آخر really بزنید fy

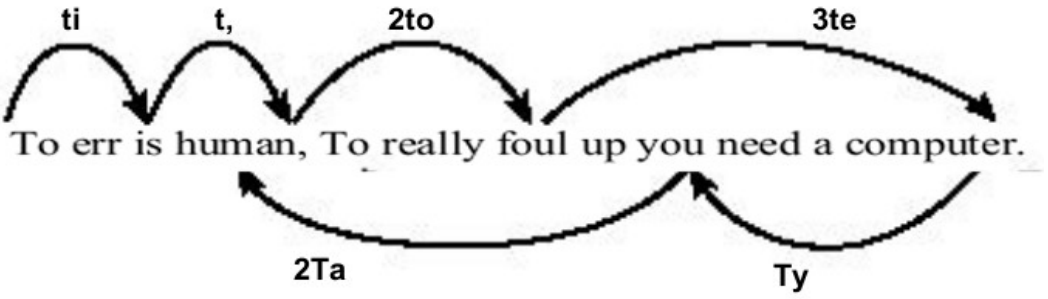
To err is human. To really foul up you need a computer.

شما می‌توانید از عدد استفاده کنید مثلا 5f < space > شما را پنج فاصله جلو می‌برد.(این کار را با 5w که پنج کلمه جلو می‌برد اشتباه نکنید ممکن است بین دو کلمه چند فاصله باشد)

To err is human. To really foul up you need a computer.

اگر از F (بزرگ) استفاده کنید جستجو به چپ انجام می‌شود. تصویر را ببینید.
دستور t هم مانند f است فقط کرسر رو حرف قبلی قرار می‌دهد(مثلا برای human روی space می‌رود). برعکس این دستور هم T (بزرگ) است. تصویر را ببینید (تصویر مفهوم نیست خودتون امتحان کنید ;-))





اگر وقتی خواستید جستجو کنید فهمیدید که اشتباه کرده‌اید مثلا به جای f زده‌اید F می‌توانید با <esc> کار خود را کنسل کنید(در ویم بیشتر کارها با <esc> کنسل می‌شوند.)

رفتن به یک سطر خاص

اگر به c یا c++ برنامه نوشته باشید یا این ارور اشنایید.

prog.c:3: 'j' undeclared (first use in this function)

این ارور می‌گوید که در سطر سوم مشکلی هست.

یک راه این است که بزنید 999k بعد 3j (فصل اول یادتان هست j سطر پایین k سطر بالا) این کار کمی سخت است.
بهتر است از G استفاده کنید
3G شما را به سطر سوم می‌برد
1G شما را به سطر اول و G یا 0G شما را به سطر آخر می‌برد.

به من بگو کجای فایلم؟

شاید دوست دارید بدانید هر لحظه کجای فایل هستید.
یک راه این است که گذاشتن شماره سطر را فعال کنید.پس از این دستور استفاده کنید.

:set number

شماره خط یک گزینه بولین است شما می‌توانید آن را با این دستور غیر فعال کنید.

:set nonumber

به من بگو کجایم؟

دستور CTRL+G به شما در مورد محلتان اطلاعاتی ارائه می‌کند مثلا

"c02.txt" [Modified] line 81 of 153 —52%— col 1

می‌گوید شما در حال ویرایش فایل c02.txt هستید. فایل را تغییر داده‌اید شما در سطر ۸۱ از ۱۵۳ سطر حدودا ۵۲ درصد فایل و در ستون اول هستید.

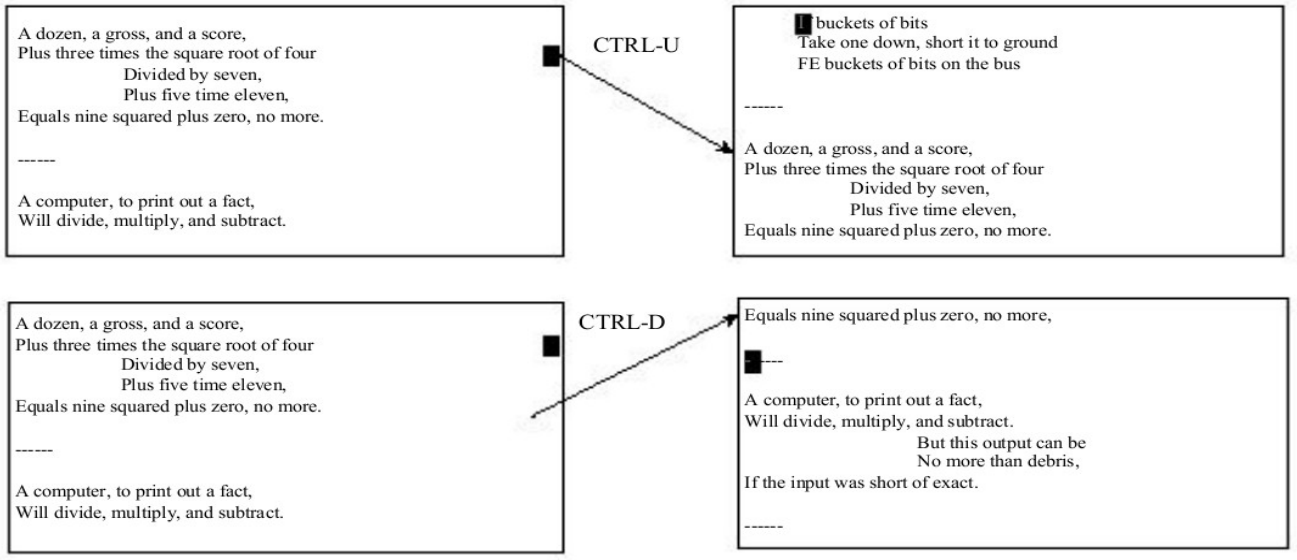
نکته:دستور نشان دادن شماره سطر فقط یک اطلاع رسانی است و به فایل کاری ندارد(تغییر نمی‌دهد)

Scrolling Up and Down

برای بالا رفتن از CTRL+U استفاده کنید (u=up) شما با این کار به اندازه نصف صفحه بالا می‌روید.

یعنی به اندازه نصف صفحه از بالا را می‌بینید(کمی گیج کننده است)

دستور CTRL+D شما را نصف صفحه پایین می‌برد.



حذف متن

همانطور که در فصل قبل دیدید با dd می‌توان کل سطر را حذف کرد.

بدانید که با dw هم می‌توان یک کلمه را حذف کرد.

شما با w به عنوان یک دستور حرکتی آشنا شدید پس می‌توان نتیجه گرفت که دستور d با هر حرکتی که همراه شود از جایی که هستیم تا مقصد را حذف می‌کند.

دستور 3w که یادتان هست پس d3w می‌تواند سه کلمه را حذف کند (شما می‌توانید از 3dw هم استفاده کنید)

دستور $ شما را به آخر سطر می‌برد پس d$ تا آخر سطر را حذف می‌کند.

3dw و d3w

دو دستور d3w و 3dw هر دو سه کلمه را پاک می‌کنند

اگر شما کنجکاو شدید بدانید که d3w سه کلمه را یکجا حذف می‌کند و 3dw سه بار یک لغت را حذف می‌کند. این فرقی است که در نتیجه تاثیری ندارد.

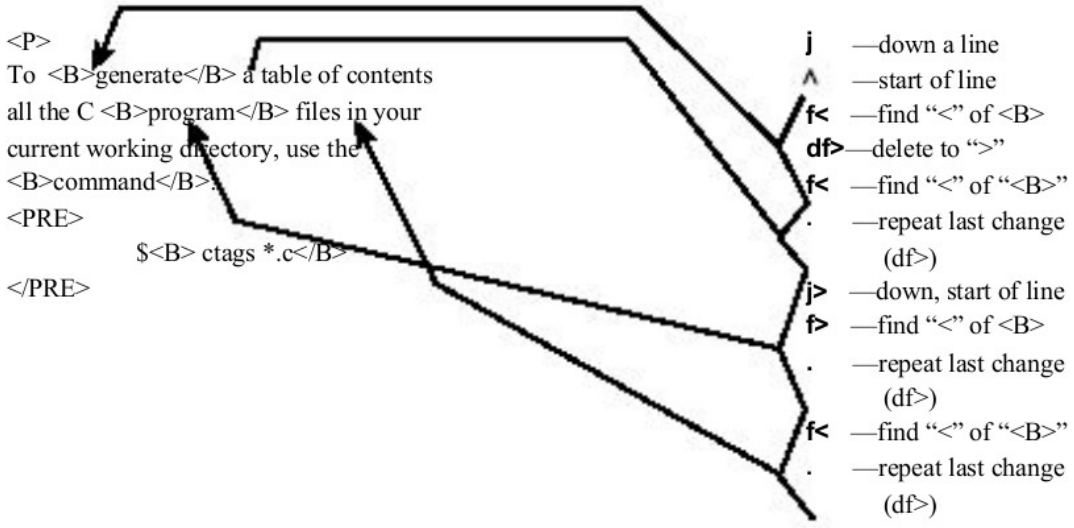
شما می‌توانید به طور همزمان دو شمارنده قرار دهید مثلا 3d2w سه بار دو کلمع را پاک می‌کند. یعنی شش کلمه را حذف می‌کند.

تغییر متن

برای تغییر متن از c استفاده کنید این دستور مثل d عمل می‌کند فقط به محض پاک کردن متن شما را در insert mode قرار می‌دهد.

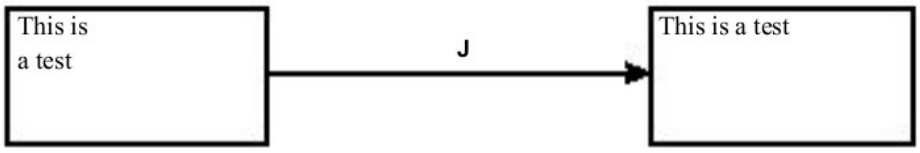
دستور . (نقطه)

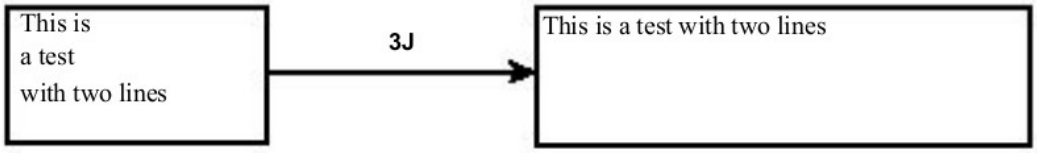
این یک دستور فوق‌العاده در ویم است.
شما می‌توانید با این دستور آخرین تغییری که داده‌اید دوباره تکرار کنید.
مثلا شما در حال ویرایش یک کد HTML هستید و می‌خواهید همه‌ی تگ‌های را حذف کنید. کرسر را روی اولین برید و آن را با دستوری مثل < df حذف کنید. حالا روی > از بعدی بروید و نقطه را بزنید نقطه آخرین دستوری که وارد کردید را اجرا می‌کند.
دراین مورد < df است



یک خط کردن

شما می‌توانید با استفاده از J (بزرگ) دو خط را به هم بچسبانید(join) (خط فعلی و بعدی)
البته بینشان یک space هم اضافه می‌کند
این دستور عدد می‌پذیرد که تعداد خطوط که باید به هم بچسبند مشخص می‌کند کوچکترین عدد ۲ است





جابجا کردن حرف

دستور rx حرفی که زیر کرسر است را با x عوض می‌کند
با وارد کردن 5rx پنج حرف از جایی که کرسر هست با x عوض می شود.

This iz a test. rs This is a test.
This is a test. 5ra aaaaais a test.

نکته: این دستور با <enter> هم جواب می‌دهد یعنی< r<enter به جای حرف یک enter می‌گذارد
همچنین دستور < 5r<enter جای پنج حرف یک enter می‌گذارد

Changing Case

با استفاده از دستور ~ می توانید حرفی که کرسور روی آن است تغییر دهید بزرگ به کوچک و کوچک به بزرگ – شما می‌توانید از چیزی مثل ~5 هم استفاده کنید که پنج حرف را تغییر case می‌دهد

now is the time. . . . ~ Now is THE time. . . .
now is The time. . . . 14~ NOW IS the TIME. . . .

ماکرو

شما با دستور دات آشنا شدید ولی اگر بخواهید کاری پیچیده‌تر کنید باید از ماکرو استفاده کنید
شما می‌توانید با دستور qcharacter شروع به ذخیره یک ماکرو می کند.
The qcharacter command records keystrokes into the register named character
این حرف باید بین a و z باشد.
ببینید چگونه می‌توان از ماکرو استفاده کرد.
شما لیستی از نام چند فایل دارید

stdio.h
fcntl.h
unistd.h
stdlib.h

شما دوست دارید اینجوری شود

#include “stdio.h”
#include “fcntl.h”
#include “unistd.h”
#include “stdlib.h”

برای این کار شما می‌توانید از ماکرو استفاده کنید
روی اولین حرف از اولین خط بروید حالا دستورات زیر را وارد کنید

qa

با این کار ذخیره ماکرو در رجیستر a شروع می‌شود

^

برو اول خط

i#include”<esc>

این عبارت را اول خط وارد می‌کند

$

برو آخر خط

a”<esc>

این عبارت را در انتهای خط اضافه کند(append کند)

j

برو به سطر بعدی

q

توقف ذخیره ماکرو

حالا می‌توانید با a@ از این ماکرو استفاده کنید
شما می‌توانید از عدد هم برای نام ماکرو استفاده کنید

```
stdio.h
fcntl.h                          شروع
unistd.h
stdlib.h

#include “stdio.h”               اینجا ماکرو را ذخیره کردیم
fcntl.h
unistd.h
stdlib.h


#include “stdio.h”
#include “fcntl.h”               a@ اجرای ماکرو
unistd.h
stdlib.h

#include “stdio.h”
#include “fcntl.h”
#include “unistd.h”              اجرای ماکروی 2@a
#include “stdlib.h”
```

کاراکترهای ویژه

بعضی از کارکترها روی کیبورد نیست مثلا ©
برای اینکار از دیگراف استفاده کنیدمثلا برای © در insert modeوارد کنید CTRL+K Co
برای مشاهده همه دیگراف‌های موجود وارد کنید
:digraphs

اخطار:
The digraphs are set up assuming that you have a standard ISO-646 character set. Although this is an international standard, your particular display or printing system might not use it.

```
~
:digraphs
~! ¡ 161    c| ¢ 162    $$ £ 163    ox ¤ 164    e= ¤ 164    Y– ¥ 165
||  ¦ 166    pa § 167    "" ¨ 168    cO © 169    a- ª 170    << « 171
–, ¬ 172    — – 173    rO ® 174    –= ¯ 175    ~o ° 176    +– ± 177
22² 178    33³ 179    ´´ ´ 180    ju µ 181    pp ¶ 182    ~. • 183
,, ¸ 184    11¹ 185    o– º 186    >> » 187    14¼ 188    12½ 189
34¾ 190    ~? ¿ 191    A` À 192    A´ Á 193    A^ Â 194    A~ Ã 195
A" Ä 196    A@ Å 197    AA Å 197    AE Æ 198    C, Ç 199    E` È 200
E´ É 201    E^ Ê 202    E" Ë 203    I` Ì 204    I´ Í 205    I^ Î 206
I" Ï 207    D- Ð 208    N~ Ñ 209    O` Ò 210    O´ Ó 211    O^ Ô 212
O~ Õ 213    O" Ö 214    /\ × 215    OE × 215    O/ Ø 216    U` Ù 217
U´ Ú 218    U^ Û 219    U" Ü 220    Y´ Ý 221    Ip p 222    ss ß 223
a` à 224    a´ á 225    a^ â 226    a~ ã 227    a" ä 228    a@ å 229
aa å 229    ae æ 230    c, ç 231    e` è 232    e´ é 233    e^ ê 234
e" ë 235    i` ì 236    i´ í 237    i^ î 238    i" ï 239    d- ð 240
n~ ñ 241    o` ò 242    o´ ó 243    o^ ô 244    o~ õ 245    o" ö 246
:- ÷ 247    oe ÷ 247    o/ ø 248    u` ù 249    u´ ú 250    u^ û 251
u" ü 252    y´ ý 253    ip p 254    y" ÿ 255
Press RETURN or enter command to continue
```

جستجو

در این فصل راه‌های جستجو در متن را یاد می‌گیرید. دستورات جستجو در ویم نسبتا ساده‌اند پس شما می‌توانید آن‌ها را زود یاد بگیرید.

در این فصل این‌ها را یاد می‌گیرید:

–جستجوی ساده به جلو

–تنظیمات جستجو

–Incremental Searches (نمی دونم معادل فارسیش چیه!)

–تغییر مسیر

–Basic Regular Expressions

جستجوی ساده

برای این کار از دستور / استفاده کنید. مثلا اگر به دنبال لغت include می‌گردید بزنید

```
/include
```

(البته شما برای این دستور باید یک <enter> هم بزنید)

نکته: بعضی حروف برای ویم معنای خاصی دارند

```
.*[]^%/\?~$
```

پس برای استفاده از آن‌ها باید قبلشان یک \ قرار دهید مثلا

```
/\.
```

که به دنبال نقطه می‌گردد.

(اگر به کلیدهای کیبورد دقت کنید / و ? یک جا هستند- برای این که به یاد داشته باشید جستجو کجاست خوبه)

برای پیدا کردن include بعدی لازم نیست دستور قبلی را کامل بزنید

```
/<enter>
```

راه دیگر هم استفاده از دستور n است فقط بزنید n تا include بعدی را نشان دهد.

تاریخچه جستجوها

جستجوی ویم دارای قابلیت نگهداری تاریخچه (history) است.مثلا شما این سه جستجو را انجام داده‌اید.

```
/one
/two
/three
```

حالا یک جستجو انجام دهیم

وارد کنید / و به جای این‌که <enter> کنید بالا را بزنید ویم در خط فرمانتان

/three

را قرار می‌دهد اگر <enter> کنید ویم به دنبال three می‌گردد.

اگر به جای <enter> کردن دوباره بالا را بزنید جای three را two می‌گیرد و اگر دوباره بالا را بزنید one جایگزین two می‌شود

به طور خلاصه پس از این‌که چند جستجو انجام دادید می‌توانید با <up> و <down> یکی از جستجوهای قبلی را انتخاب کنید

تنظیمات جستجو

شما برای جستجو تنظینات زیادی دارید که در اینجا تعدادی را می‌بینید

highlighting

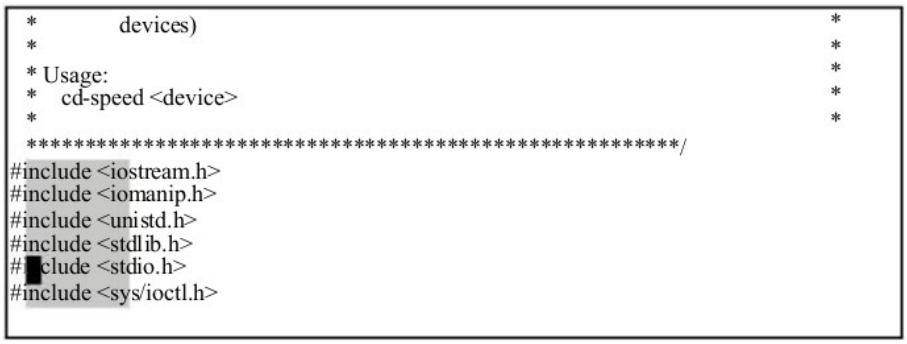
در صورتی که این تنظیم فعال باشد ویم تمام رشته‌های یافت شده در متن را های‌لایت می‌کند

:set hlsearch

یا

:set hls[1]

مثلا وقتی به دنبال include بگردید همه‌ی include های متن های‌لایت می‌شوند



برای غیر فعال کردن این قابلیت از این دستور استفاده کنید

:set hls

برای این که فعلا hl را غیر فعال کنید از دستور

:nohls

استفاده کنید با این کار نتایج جستجوی قبلی به شکل عادی بر می‌گردند ولی خود hl-غیر فعال نشده(یعنی در جستوی بعدی باز هم رشته‌های پیدا شده های‌لایت می‌شوند)

Incremental Searches

شما برای جستجو متن را تعریف می‌کنید و سپس ویم به دنبال آن می‌گردد خوب شاید دوست داشته باشید همانطور که متن را وارد

1 .این مخفف دستور قبلیه. من مخفف بیشتر دوست دارم برای همین کمتر دستور را به طور کامل می‌گذارم

می‌کنید ویم جستجو کند

برای فعال کردن از قابلیت از دستور زیر استفاده کنید

:set incsearch

یا

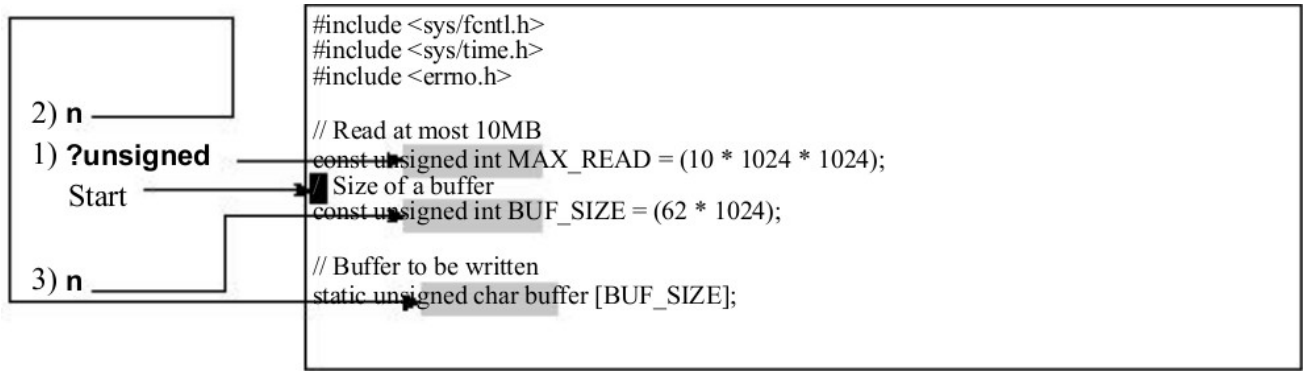
:set is

جستجو به عقب

برای این کار به جای / از ? استفاده کنید

دستور n جستجوی قبلی را تکرار می‌کند

یعنی اگر جستجوی قبلی رو به عقب بود n به عقبتر می‌رود و اگر به جلو بود n جلوتر می‌رود



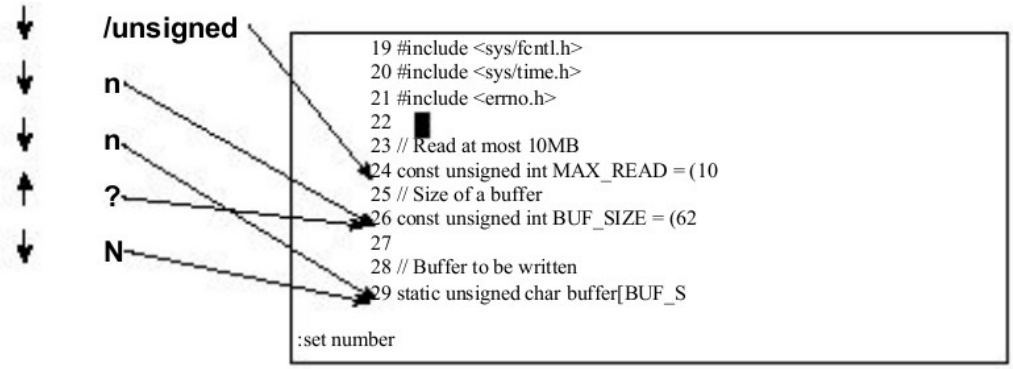
تغییر جهت

اگر شما دستور

/unsigned

را برای جستجوی به جلو زده‌باشید. برای تغییر جهت جستجو می‌توانید از دستور ? استفاده کنید. دستور n نیز در جهت جدید جستجو می‌کند.

دستور N در خلاف جهت فعلی یک حرکت می‌کند.



Basic Regular Expressions

ویم برای جستجو از ابزار منظمی استفاده می‌کند. ابزار جستجو به شدت قوی هستند و راه جمع و جوری برای یک جستجو هستند.
خوب است از چند ابزار ساده استفاده کنیم

آغاز ^ و پایان $ خط

اگر به دنبال include بگردید نتیجه هر include را در هر جای صفحه نشان می‌دهد
خوب شما ممکن است فقط به دنبال آن include ی می‌گردید که در ابتدای خط است

```
/^include
```

(این دستور با ^ که پیشتر گفتم تداخل ندارد)
شاید هم به دنبال یک کلمه در انتهای خطوط بگردید

```
/was$
```

هر حرفی

ممکن است شما به دنبال کلمه‌ای می‌گردید که حرف اول آن c و حرف سوم آن m باشد ولی حرف دوم آن را فراموش کرده‌اید

```
/c.m
```

بله! به جای حرف فراموش شده از نقطه استفاده کردم

کاراکترهای ویژه

شما کاربرد چند کارکتر ویژه را دیدید
برای استفاده از خود این کارکترها (مثلا نقطه) قبل از آن‌ها یک \ قرار دهید مثلا

```
/the\.
```

که به دنبال

```
the.
```

می‌گردد.

تکه متن و چند فایلی

در این فصل یاد می‌گیرید چگونه با قطعه‌های بزرگتر متن کار کنید تا کارهای مثل کپی و پیست انجام دهید.

با بیشتر ویرایشگرها می توانید کپی و پیست کنید ولی ویم مجهز به یک لیست هم هست. شما می‌توانید چندین چیز برای کپی و پیست کردن نگه دارید. در حالی که بیشتر ویریشگرها به یک چیز در کلیپ‌برد محدودند ولی در ویم کلیپ بورد شما بیش از ۲۶ فضا دارد.

تا کنون یاد گرفتید چگونه با یک فایل کار کنید در این فصل یاد می‌گیرید با چندین فایل کار کنید و بین آنها کپی و پیست کنید.

در این فصل یاد می‌گیرید

–کات,کپی و پیست کردن ساده

–نشانه گذاری متن

–کپی کردن متن در رجیستر و استفاده از yank

–Filtering text

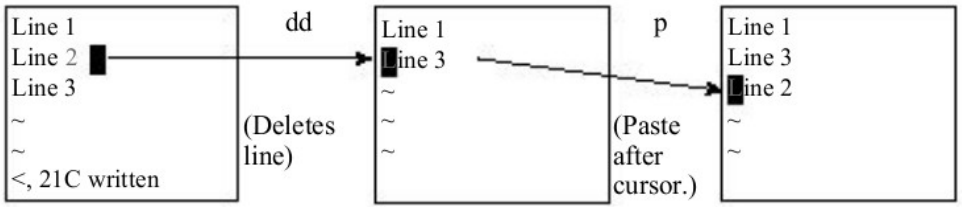
–ویرایش چند فایل

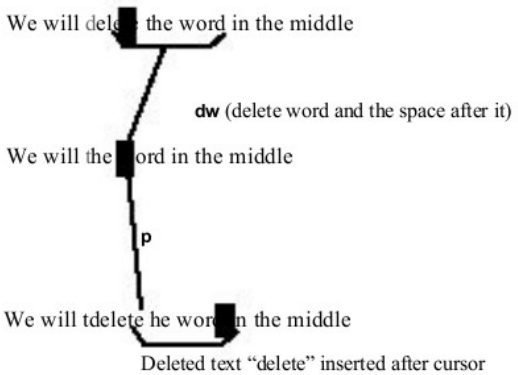
کات کپی و پیست

وقتی شما متنی را پاک می‌کنید- با دستور x, d یا هر چیز دیگری این متن ذخیره می‌شود و شما می‌توانید آن را با p پیست کنید (اول put – اگر فکر کردید paste پس فکرتون رو عوض کنید!)

ببینید چطور کار می‌کند! اول کل خط را با dd پاک کردیم حالا به جایی که می‌خواهیم متن را قرار دهیم می‌رویم و p. متن در خط بعد از کرسر قرار گرفت چون شما یک خط را کامل پاک کردید ویم یک خط جدید ایجاد می‌کند.

اگر شما بخش کوچکی را پاک کنید مثلا با dw موقع put کردن متن را دقیقا بعد از کرسر قرار می‌دهد.



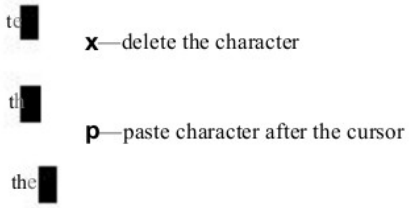


کارکتر بازی

بعضی وقتها وقتی تایپ می‌کنید انگشتان شما سریعتر از مغزتان عمل می‌کنند مثلا به جای the می‌نویسید teh

ویم کار شما را راحت کرده! روی e بروید و آن را با x پاک کنید حالا p را بزنید تا کارکتر بعد کرسر قرار بگیرد.

xp



بیشتر در مورد put

شما می‌توانید چندد بار p بزنید و هر بار یک کپی دیگر وارد متن کنید.

دستور p متن را بعد از کرسر می‌گذارد و P آن را قبل.

این دستور عدد هم می‌گیرد مثلا 5p

علامت گذاشتن

شما می‌توانید هر قسمتی از متن را علامت گذاری کنید. دستور ma متن زیر کرسر را به اسم a نشانه گذاری کنید. شما می‌توانید ۲۶ نشانه قرار دهید(a تا z) حتی از عدد هم می‌توانید استفاده کنید.

برای رفتن به جایی که علامت گذاری کردید از این دستور استفاده کنید.

'mark

مثلا برای نشانه قبلی که درست کردید بزنید 'a (کوتیشن تکی رو هم بگذارید)

این دستور برای پاک کردن یک متن طولانی مفید است

–به ابتدای متنی که می‌خواهید پاک شود بروید

–با دستور ma آن را مارک کنید. (این دستور به نام a نشانه گذاری می‌کند)

–به انتهای متنی که باید پاک شود بروید

–از اینجا تا جایی که به اسم a علامت گذاشتید پاک کنید d'a

نکته: شما مجبور نیستید از a استفاده کنید هر حرفی بین a-z آزاد است.

نکته: شما مجبور نیسیتد ابتدای متن را مارک کنید از انتها هم می‌شود

نکته: انچه مارک می‌کنید در متن می‌ماند یعنی اگر متن را جاجا کنید مارک هم جابجا می‌شود (اگر متن را پاک کنید -مثلا با d- مارک هم از بین می‌رود)

نشانه‌ها کجایند؟

برای مشاهده همه مارک‌ها بزنید

:marks

شما چند مارک از پیش تعیین شده هم می‌بینید

' شما را به جای قبلی کرسر می‌برد

" شما را به محلی که فایل در آخرین بار آنجا بسته شده می‌برد

] به ابتدای آخرین جایی که اینسرت کردید

[به انتهای آخرین جایی که اینسرت کردید

برای مشاهده‌ی مارک‌هایی که تعریف کردید بزنید

:marks args

برای مشاهده اطلاعات یک مارک خاص به جای args اسمش را وارد کنید

```
 *         the data from an input
 * (.c) file.
 */
struct in_file_struct {
:marks
mark  line   col    file/text
 '          67      0   *^I^I^I   into the "bad" list^I^I*
 a           1      0   #undef USE_CC^I/* Use Sun's CC com
 b           8      1   * Usage:^I^I^I^I^I^I*
 c          14      1   *^I^I^I   (default = proto_db)^I^I
 d          25      1   *^I—quote^I^I^I^I^I^I*
 "           1      0   #undef USE_CC^I/* Use Sun's CC com
 [         128     42   * in_file_struct — structure that
 ]         129     12   *         the data from an input
Press RETURN or enter command to continue
```

yanking (کپی کردن)[8]

سالها برای کپی کردن متن از یک راه ساده استفاده می‌کردم. با دستور d ان را پاک و با p دوباره همانجا قرارش می‌دادم بعد به محل مورد نظر می‌رفتم و دوباره p

راه دیگری هم هست. دستور y متن را به یک رجیستر می‌کشد بدون این پاک شود

این دستور کاملا مثل d کار می‌کند مثلا yy کل خط را می‌کشد

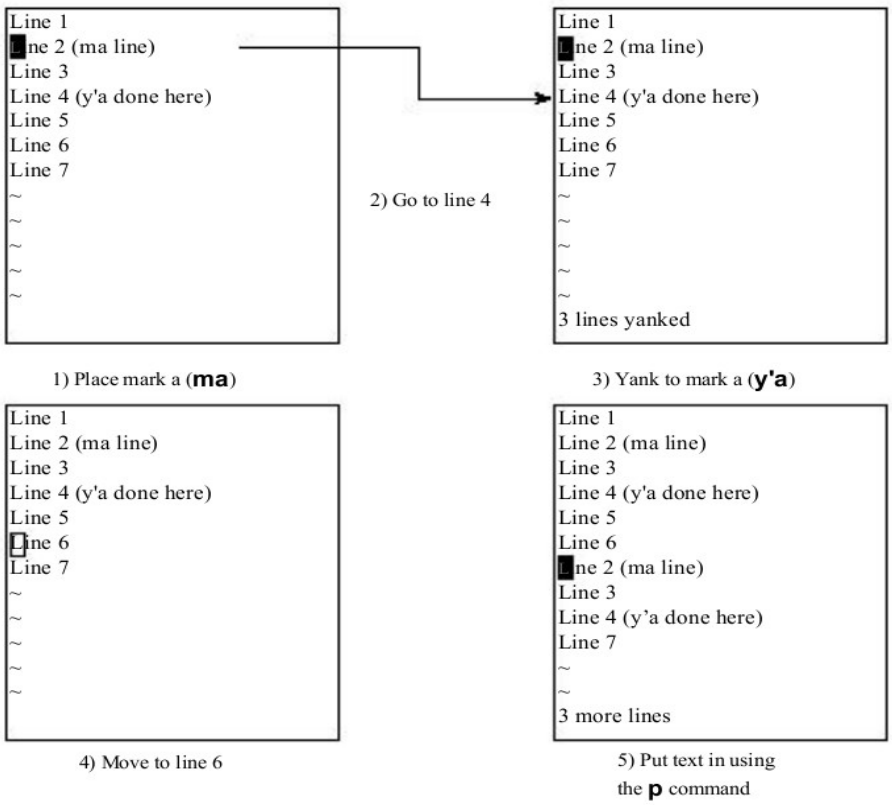
بیشتر ویرایشگرها این کار را کپی نامیده‌اند

دستور Y یک خط را کامل کپی می‌کند.

اگر با عدد باشد چند خط بعد هم کپی می‌شوند

پس حدس می‌زنید دستور D و C چه می‌کند)

8 معنیش می‌شه زود کشیدن



فیلتر

دستور

!motion

دستور سیستمی motion را روی قطعه مورد نظر اجرا و آن را با خروجی جایگزین می‌کند.

ممکن است کمی گیج کننده باشد. مثالی می‌زنم. دستوری به نام sort هست که فایل نامرتب را مرتب می‌کند و با دستور زیر در خط فرمان اجرا می‌شود

$ sort <input.txt > output.txt

خوب حالا چگونه از این دستور در ویم استفاده کنیم؟

مثلا شما می‌خواهید سطر اول تا دهم را مرتب کنید. به سطر اول بروید و دستوررا وارد کنید

!10G

! میگوید که شما می‌خواهید یک دستور فیلتر اجرا کنید. ویم انتظار دارد که یک دستور حرکت وارد کنید تا بفهمد دستور در چه منطقه‌ای از متن اجرا شود. از 10G می‌فهمد فیلتر باید روی جایی که کرسر هست تا سطر دهم اجرا شود. کرسر به پایین صفحه می‌پرد تا دستور مورد نظر را وارد کنید.

!10Gsort<enter>

دستور sort اجرا شده و خروجی آن جایگزین سطر اول تا دهم می‌شود

دستور !! دستور مورد نظر را روی سطر فعلی اجرا می‌کند.

مثلا شما در حال نوشتن یک فایل readme هستید و مایلید لیست فایل‌های موجود در پوشه را وارد کنید

!!ls

(برای نسخه تحت ویندوز بزنید dir)

یک ترفند جالب وارد کردن تاریخ است

!!date

ویرایش یک فایل دیگر

اگر کار شما با فایل فعلی تمام شد و می‌خواهید روی دیگر کار کنید یک راه این است که ویم را ببندید و دوباره باز کنید. راه دیگر هم دستور زیر است

:vi file

این دستور فایل فعلی را می‌بندد و file را باز می‌کند. اگر تغییرات را ذخیره نکرده باشید خطایی می‌بینید

No write since last change (use ! to override)

شما چند راه حل دارید مثلا ذخیره کنید یا این که بدون ذخیره کردن فایل را ببندید

:vi! File

به جای vi می‌توانید از e هم استفاده کنید.

فقط خواندنی[9]

برای این کار از دستور

:view file

استفاده کنید این دستور دقیقا مثل vi کار می‌کند. فقط شما قادر نیستید تغییرات را در فایل ذخیره کنید

(البته با !w: می توانید)

کار با چند فایل

تا اینجا یاد گرفتید چگونه روی یک فایل کار کنید. حالا روی چند فایل کار کنید.

شما با چنین دستوری می‌توانید چند فایل باز کنید.

$ gvim one.c two.c three.c

این دستور ویم را باز می‌کند و اعلام می‌کند شما در حال ویرایش سه فایل هستید. به طور پیشفرض شما در فایل اول هستید. برای رفتن به فایل بعدی بزنید

:next

اگر فایل ذخیره نشده باشد شما یک اخطار می‌بینید که چند راه حل دارد

9 Read only

فایل را ذخیره کنید

:write

و به بعدی بروید

:next

یا به طور خلاصه بزنید

:wnext

یا ذخیره نکنید

:next!

(اگر ذخیره نکنید تغییرات را از دست می‌دهید)

یک تنظیم هم هست که شما را از این گونه اخطارها راحت می‌کند. فایل به طور خودکار ذخیره شود و دیگر اخطار ندهد

:set autowrite

:set noautowrite

این دستور با عدد هم همراه می‌شود

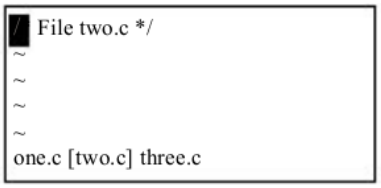
:2next

روی چه فایل‌هایی کار می‌کنم؟

دستور

:args

اسم فایل‌هایی که باز کردید نشان می‌دهد. و فایلی که فعال است را در براکت نشان می‌دهد.



رفتن به فایل قبلی

از دستور

:previous

یا

:Next

استفاده کنید.

رفتن به فایل اول و آخر

فرقی نمی‌کند در چندمین فایل هستیدفقط بزنید

:first

:last

ویرایش دو فایل

شما دو فایل باز کردید

$ gvim one.c two.c

کمی آن را ویرایش و به فایل دوم می‌روید

:wnext

حالا ویم فایل قبلی را به عنوان یک فایل نوبتی می‌شناسد. شما می‌توانید با CTRL+^ به آن فایل بروید.

با زدن CTRL+^ شما به فایل ۱ می‌روید و حالا فایل ۲ فایل نوتبی می‌شود

نکته:اگر از همان ابتدا CTRL+^ می‌زدید ارور می‌گرفتید چون فایل نوبتی نداریم یک بار باید :next کنید

پنجره‌ها

شما تا کنون روی یک پنجره کار کردید. در این فصل شما با شکافتن صفحه به چندین پنجره می‌توانید چندین فایل را همزمان ویرایش کنید. در این فصل با بافر هم آشنا می‌شوید.

A buffer is a copy of a file that you edit along with the setting and marks that go with it.

–باز کردن یک پنجره جدید

–انتخاب یک پنجره

–ویرایش چند فایل همزمان

–تغییر سایز پنجره

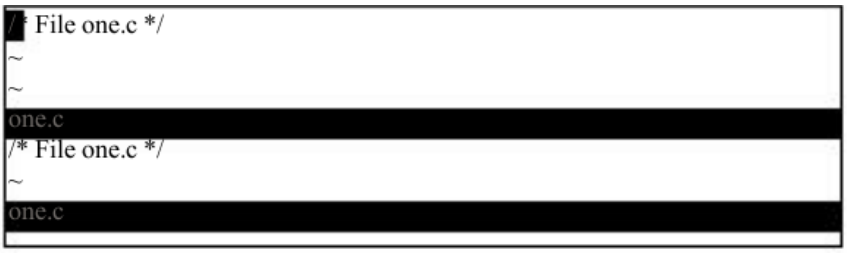
–استفاده ساده از بافر

باز کردن یک پنجره جدید

راحت‌ترین راه برای باز کردن یک پنجره جدید دستور زیر است

:split

این دستور یک پنجره به صفحه اضافه می‌کند و صفحه را بین این دو پنجره تقسیم می‌کند.



این دو پنجره هر دو یک فایل را نشان می‌دهند پس شما می‌توانید همزمان چند بخش فایلتان را ببینید.

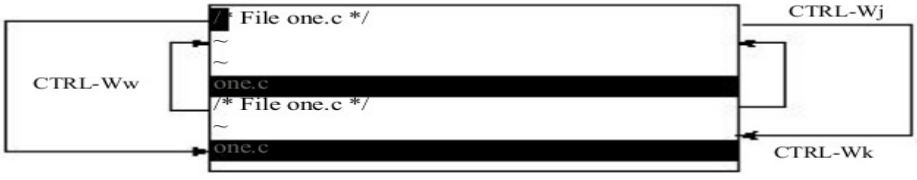
دستور CTRL-Ww شما را به پنجره پایینتر می‌برد و اگر در پایینترین پنجره باشید به بالاترین پنجره می‌روید.

همچین دستور CTRL-Wj شما را به پنجره پایینتر و CTRL-Wk به پنجره بالاتر می‌برد.

در نسخه گرافیکی می‌توانید روی پنجره مورد نظر کلیک کنید.

برای بستن یک پنجره از ZZ یا دستور زیر استفاده کنید

:q



باز کردن پنجره با فایل دیگر

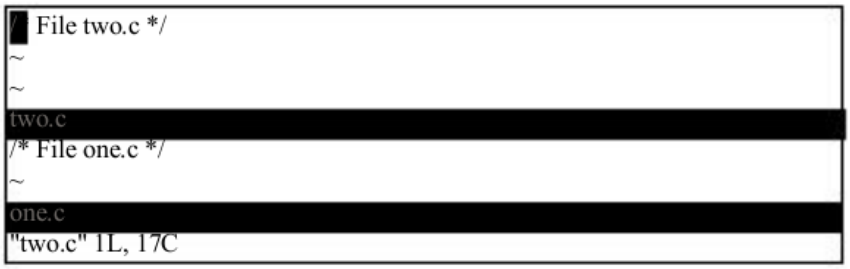
دستور

:split file

پنجره جدیدی باز می‌کند و ویرایش فایل داده شده را شروع!

مثلا ببینید این دستور چه می‌کند

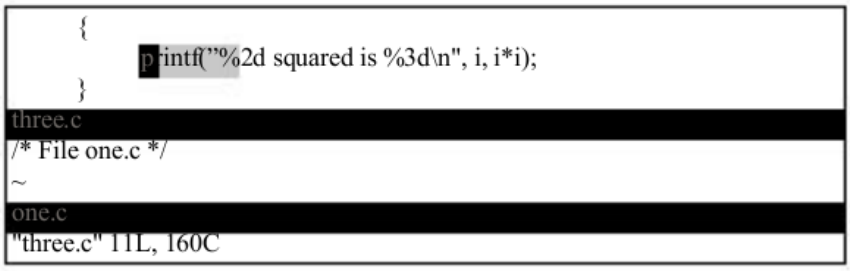
:split two.c



split می‌تواند یک دستور را که به صورت command+ تعریف شده باشد حین باز کردن فایل اجرا کند. مثلا

:split +/printf three.c

حین باز کردن فایل سوم در آن دنبال printf می‌گردد.



کنترل سایز پنجره

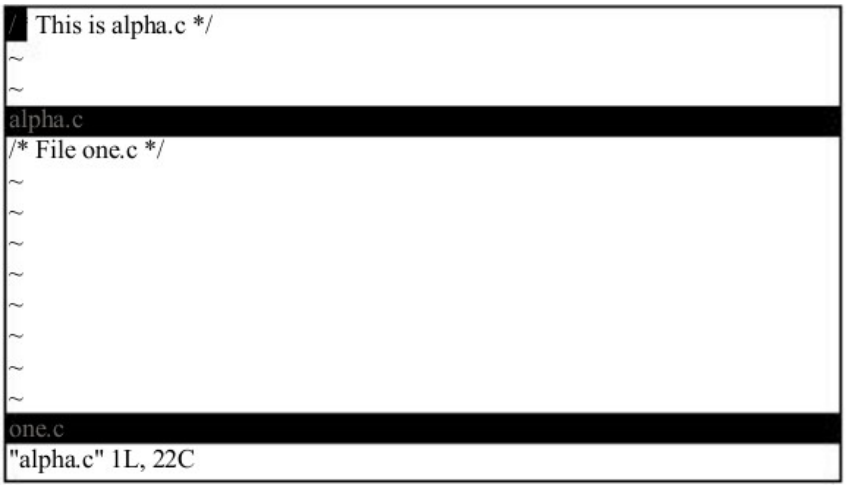
دستور split می‌تواند با عدد همراه شود. عدد نشان دهنده تعداد خطوط برای فایل جدید است.

مثلا این دستور فایل آلفا را در پنجره جدیدی به اندازه ۳ خط باز می‌کند.

:3 split alpha.c

از فاصله برای روشن‌تر بودن استفاده کردم. بدون آن هم کار می‌دهد!

:3split alpha.c



خلاصه

:count split +command file

count

سایز پنجره جدید براساس تعداد خط (پیشفرض به دو قسمت مساوی)

+command

دستور آغازین

file

نام فایل برای ویرایش (پیشفرض فایل فعلی)

دستور new

دستور new: دقیقا مثل split: کار می‌کند با این تفاوت که دستور split صفحه را تقسیم و فایل قبلی را در پنجره جدید باز می‌کند ولی new صفحه را تقسیم و فایل جدیدی در صفحه باز می‌کند

تقسیم کردن و نگاه کردن!

شاید دوست داشته باشید حین ویرایش فایل جدید باز کنید ولی آن را تغییر ندهید.

شما می‌توانید دو دستور split: و view: را مخلوط کنید

:sview

تغییر سایز پنجره

تغیر سایز پنجره در gvim بسیار ساده است فقط با موس خط جدا کننده‌ی پنجره‌ها را بالا و پایین بکشید.

در نسخه تحت ترمینال باید از دستور استفاده کنید

countCTRL-W+

پنجره را به اندازه count بزرگتر می‌کند. (پیشفرض یک)

countCTRL-W-

پنجره را به اندازه count کوچکتر می‌کند. (پیشفرض یک)

countCTRL-W=

همه‌ی پنجره‌ها را مساوی می‌کند

countCTRL-W -

سایز پنجره جدید را به مقدار داده شده تغییر می‌دهد (پیشفرض بیشترین حد ممکن)

بافر

The Vim editor uses the term buffer to describe a file being edited

بافر یک کپی است از فایلی که ویرایش می‌کنید. وقتی ویرایش تمام شد و خارج شدید محتویات بافر در فایل ذخیره می‌شود. در بافر فقط محتویات فایل نیست, بلکه مارک‌ها و تنظیمات و چیزهای دیگری هم در بافر ذخیره می‌شود.

تعریف بافر راحت بود اگر می‌توانستم بگویم اگر در صفحه پنجره دارید آن یک بافر است اگر هم ندارید خوب نیست!

ولی مطلبی این تعریف را به هم می‌ریزد, بافر مخفی!

مثلا شما one.c را ویرایش می‌کنید, حالا کمی با two.c کار دارید. می‌توانید با split فایل دوم را باز کنید ولی شاید دوست دارید هر لحظه فقط یک فایل ببینید.

می‌توانید two.c به بیشترین حد ممکن بزرگ کنید ولی هنوز بخشی از one.c پیداست.

راه دیگر هم بستن one.c است که شما همه تغییرات را از دست می‌دهید.

راه دیگر هم دستور

:hide

است که باعث می‌شود بافر فعلی مخفی شود. این باعث می‌شود که از صفحه محو شود. ولی ویم هنوز می‌داند شما این فایل را ویرایش می‌کردید و مارک‌ها و چیزهای دیگر را جایی نگه می‌دارد.

بافر می‌تواند در سه وضعیت باشد

Active

روی صفحه نمایش داده می‌شود

Hidden

فایل ویرایش می‌شود. فقط فعلا دیده نمی‌شود

Inactive

فایل دیگر ویرایش نمی‌شود. ولی هنوز اطلاعات آن جایی هست!

غیر فعال کمی کار دارد. وقتی شما یک فایل جدید ویرایش می‌کنید و به قبلی نیاز ندارید پس ویم محتویات آن را دور می‌ریزد فقط

اطلاعات مارک‌ها و چند چیز دیگر که می‌تواند مفید باشد به نام فایل در جایی می‌مانند. همچنین فایلی که هنوز ویرایش نشده هم یک بافر غیر فعال است.

برای مشاهده لیست بافرها بزنید

:buffers

در تصویر خط اول شماره هر بافر, دوم پرچم بافر که وضعیت آن را نشان می‌هد و سومی اسم فایل وابسته به بافر.

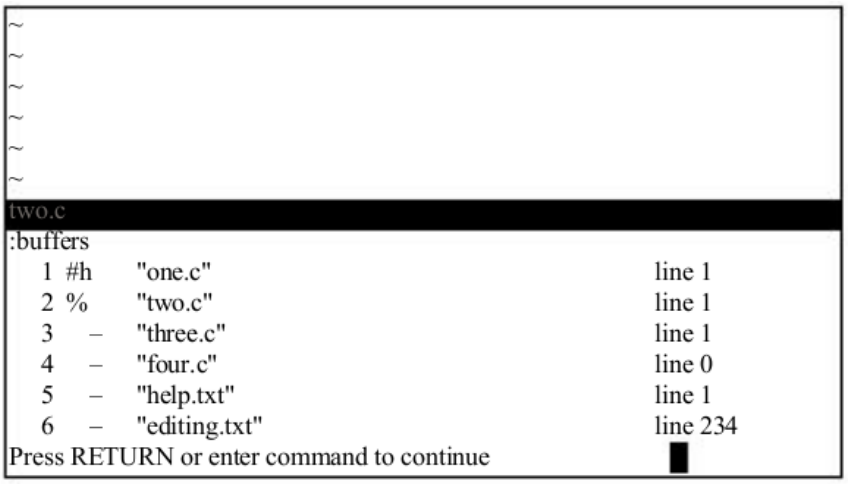
a بافر فغال

h بافر مخفی

% بافر فعلی

بافر نوبتی

+ فایل تغییر کرده



تصویر ار نسخه قدیمی ویم هست.

اینجا شش بافر می‌بینید

۱- این بافر مخفی است و چون قبل از بافر فعلی رویش کار می‌کردید بافر نوبتی است

۲- بافر فعلی (نسخه جدید ویم می‌نویسد a%)- شما در حال ویرایش این هستید

۳- بافر غیرفعال- شما می‌خواهید آن را ویرایش کنید ولی هنوز نکردید

۴- مثل ۳

۵- وقتی از دستور help: استفاده می‌کنید ویم دو فایل از می‌کند این

۶- و این

انتخاب بافر

برای این که به یک بافر بروید بزنید

:buffer number

number همان شماره بافر در جدول است.

اگر به جای شماره بافر نام فایل آن را به خاطر دارید بزنید

:buffer file

این دستور صفحه را تقسیم و ویرایش بافر داده شده را شروع می‌کند(مخلوط کردن split و بافر)

:sbuffer number

یا

:sbuffer one.c

بقیه دستورات بافر را ببینید

:bnext

برو به بافر بعدی

:count bnext

برو به چندتا بافر بعد

:count sbnext

خلاصه split: و bnext:

:count bprevious

برو به بافر قبلی (اگر با عدد باشد چنتا قبلی)

:count sbprevious

خلاصه split: و bprevious:

:count bNext

به جای bprevious: می‌توانید استفاده کنید

:count sbNext

به جای sbprevious: می‌توانید استفاده کنید

:blast

برو به آخرین بافر لیست

:sblast

خلاصه split: و blast:

:first

برو به اولین بافر لیست

:sfirst

خلاصه split: و first:

:bmodified count

برو به چندید بافری که تغغیر یافته

:sbmodified count

خلاصه split: و bmodified:

تنظیمات بافر

وقتی آخرین پنجره‌ی یک فایل بسته می‌شود بافر آن غیر فعال می‌شود.

فایل‌های که مخفی هستند غیر فعال نمی‌شوند. اگر می‌خواهید محتویات بافرهای قدیمی خود را داشته باشید از این استفاده کنید

:set hidden

ویژوال

چیزی که ویم را از ویرایشگرهای قبلی جدا می‌کند چیزی هست به اسم ویژوال مد. با این دستور می‌توانید قسمتی از متن را علامت بزنید و بعد دستوری روی آن اجرا کنید مثلا قسمتی از متن را علامت بزنید و سپس با d آن را پاک کنید.

برعکس دستورات قبلی که شما کورکورانه اجرا می‌کنید شما می‌بینید کدام قسمت متن قرار است تغییر کند.

–شروع استفاده از ویژوال مد

–کپی کردن در ویژوال مد

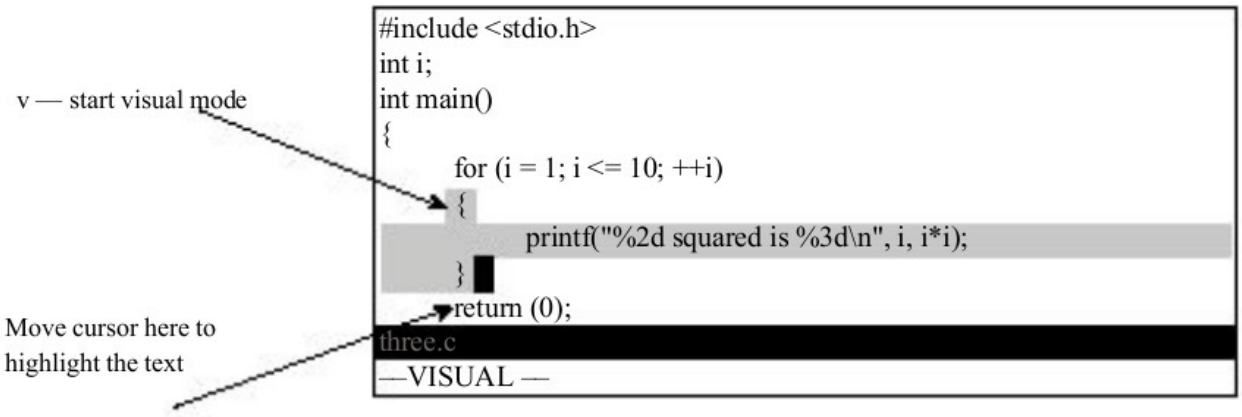
–تغییر متن در ویژوال مد

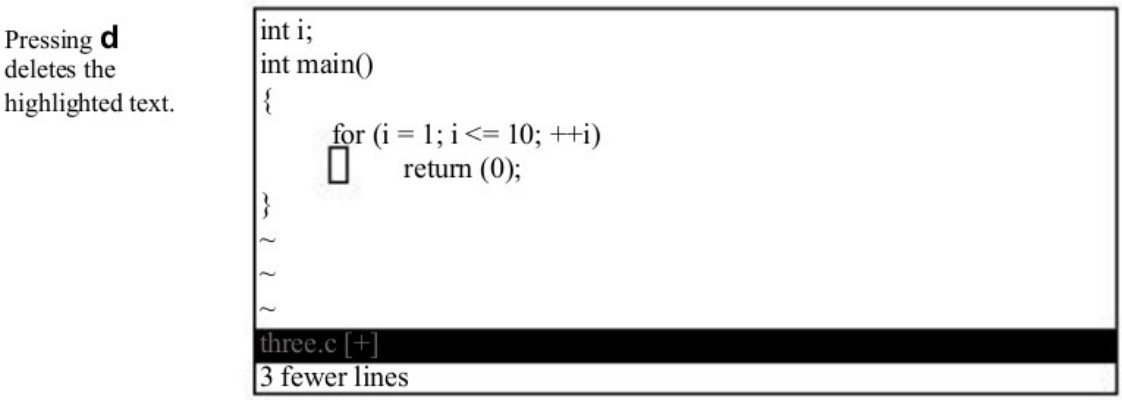
–دستور برای برنامه نویسان

–ویژوال بلاک مد

ورود به ویژوال

برای ورود به ویژوال بزنید v حالا شما وارد ویژوال مد شدید. در متن حرکت کنید تا از جایی که اول بودید تا محل فعلی علامت زده شود. حالا می‌تولنید روی متن انتخاب شده دستوری اجرا کنید, مثلا با d آن را پاک کنید.





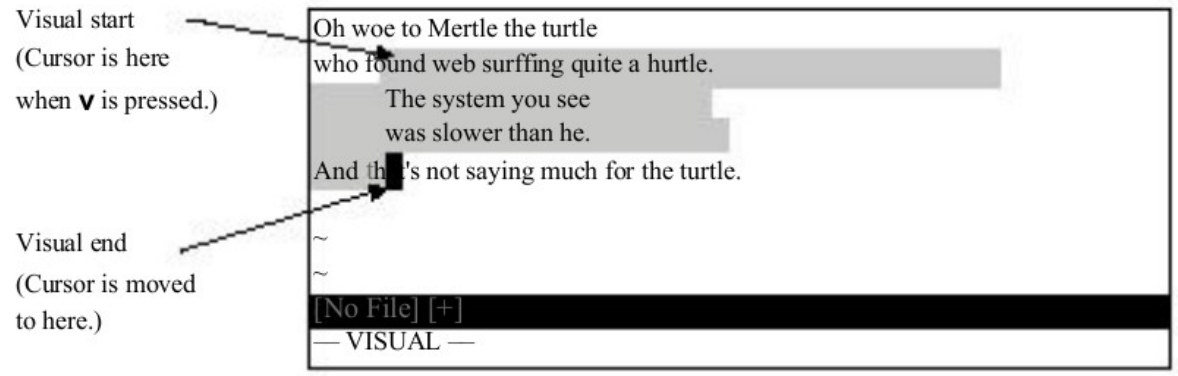
دیگر شکل‌های ویژال مد

با دستور v به صورت انتخاب حرف به حرف وارد ویژوال می‌شوید.

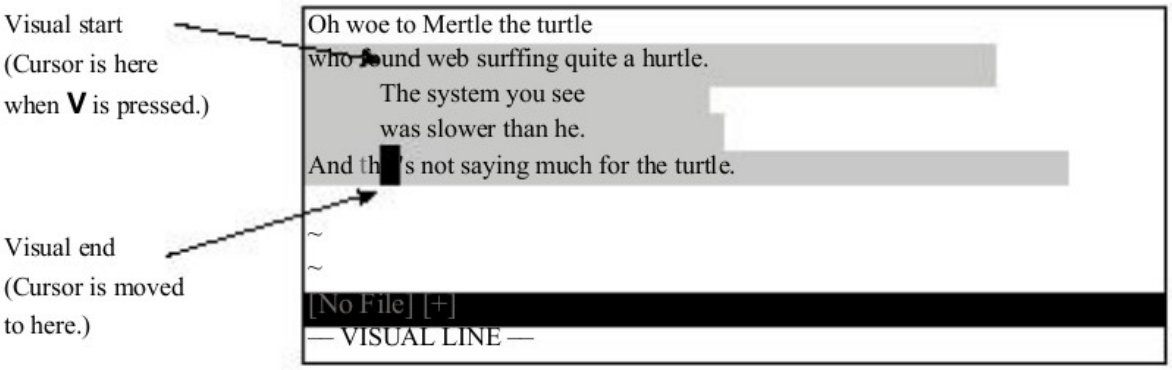
دستور V به صورت خط به خط وارد ویژوال می‌شوید. شما فقط می‌توانید خط های کامل را انتخاب کنید.

نکته: برای راهنمایی دستورات در ویژوال مد از شکل v_ استفاده کنید مثلا برای d

:help v_d

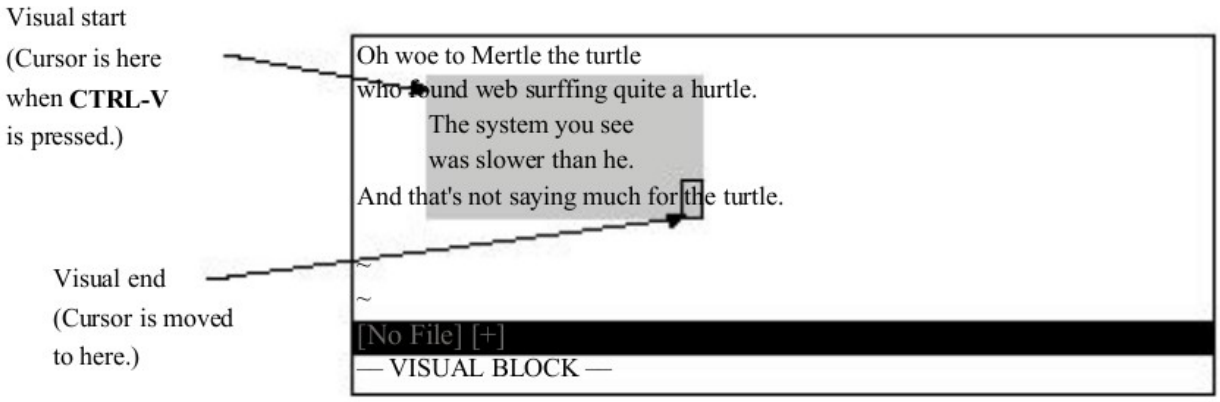


v



V

در صورتی که می‌خواهید مستطیلی از متن را انتخاب کنید با CTRL+v وارد ویژوال شوید تا مثل شکل انتخاب کنید.



خروج از ویژوال

به محض این که یک دستور ویژوال بزنید از آن خارج می‌شوید مثلا تا d را بندید شما به نورمال مد بر می‌گردید.

اگر هم نخواهید دستوری اجرا کنید, همانطور که یه یاد دارید با <esc> می‌توانید از هر وضعیتی خارج شوبد و به نورمال برگردید.

بعضی‌ها از <esc> خیلی خوششان نمی‌اید چون اگر آن را دوبار بزنید سیستم بوق می‌زند.

بار اول از ویژوال خارج می‌شود و به نورمال مد می رود و بار دوم چون در نورمال هستید اخطار می‌دهد.

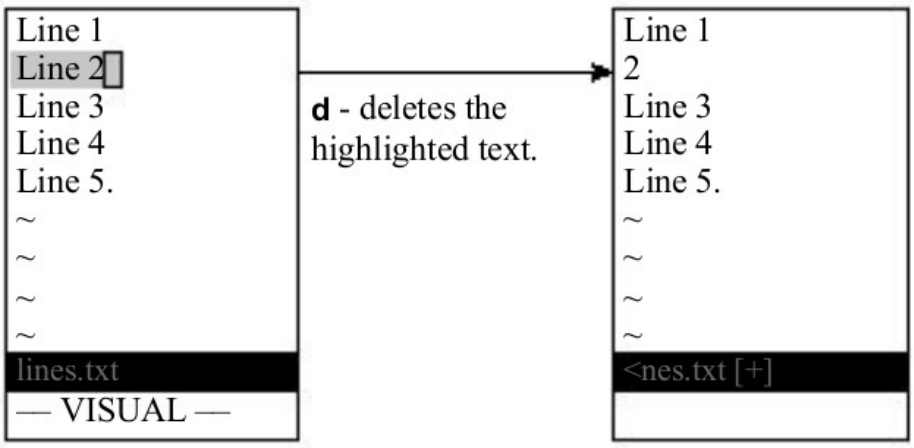
دستور CTRL+c هم همین طور عمل می‌کند.

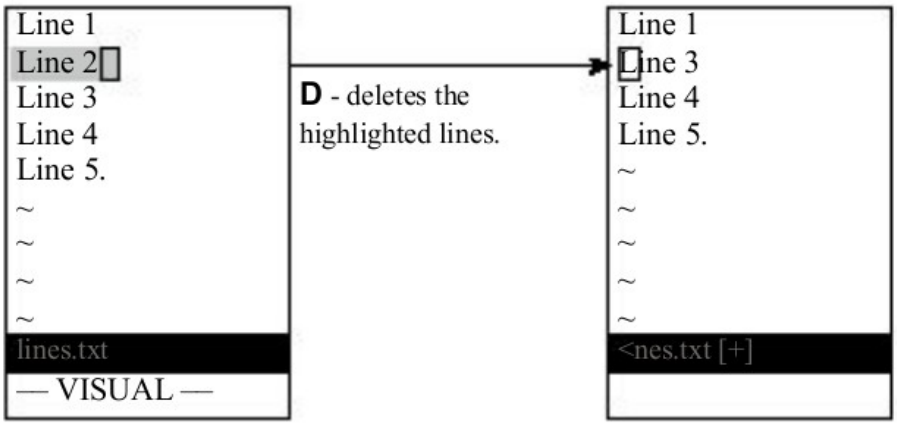
شما همچنین می‌توانید از CTRL-\CTRL-N استفاده کنید که دیگر بوق نمی‌زند.

پاک کردن متن

دستور d قسمت انتخاب شده را پاک می‌کند

دستور D خطوط انتخاب شده را پاک می‌کند حتی اگر آن خطوط کامل علامت زده نباشند.





کپی کردن

دستور y قسمت‌های علامت زده شده را در یک رجیستر کپی می‌کند
دستور Y خطوطی که علامتی داشته باشند را در رجیستر کپی می‌کند. حتی اگر آن خط کاملا علامت زده نباشد کاملا کپی می‌شود. (مثل d,D)

تغییر وضعیت

در صورتی که در یک از شکل‌های ویژوال مد هستید می‌توانید همان موقع وارد وضعیت ویژوال دیگر شوید و ویژوال فعلی را لغو کنید.
مثلا با v وارد ویژوال مد شدید حالا بزنید ctrl+v تا وضعیت خود را در ویژوال تغغیر دهید.
اگر در یکی از وضعیت‌های ویژوال مد هستید برای این که از آن خارج شوید و به نورمال مد بروید می‌توانید از <esc> استفاده کنید.
یا اینکه دوباره دستوری با آن وارد ویژوال شدید را وارد کنید.(مثلا اگر با v در ویژوال مد قرار گرفتید بزنید v تا به نرمال مد برگردید)

تغییر متن

دستور c قسمتن انتخاب شده را پاک و شما را به insert می‌برد
C خطوطی که علامت زده باشید کاملا پاک می‌کند و به insert می‌رود

چسباندن خطوط

دستور J خطوطی که علامت دار باشند به هم می‌چسباند و بین هر خط یک space می‌گذارد
اگر دوست ندارید space بگذارد از gJ استفاده کنید.

نکته:
r و s در ویژوال مشابه c کار می‌کنند. همچنین R و S مثل C.

دستوری برای برنامه نویسان

دستور < خطوط انتخاب شده را به اندازه یک shiftwidth جلو می‌برد. (تو رفتگی ایجاد می‌کند)
و > کار عکس می‌کند.
= تو رفتگی خطوط انتخاب شده را یکی می‌کند

یافتن منوال

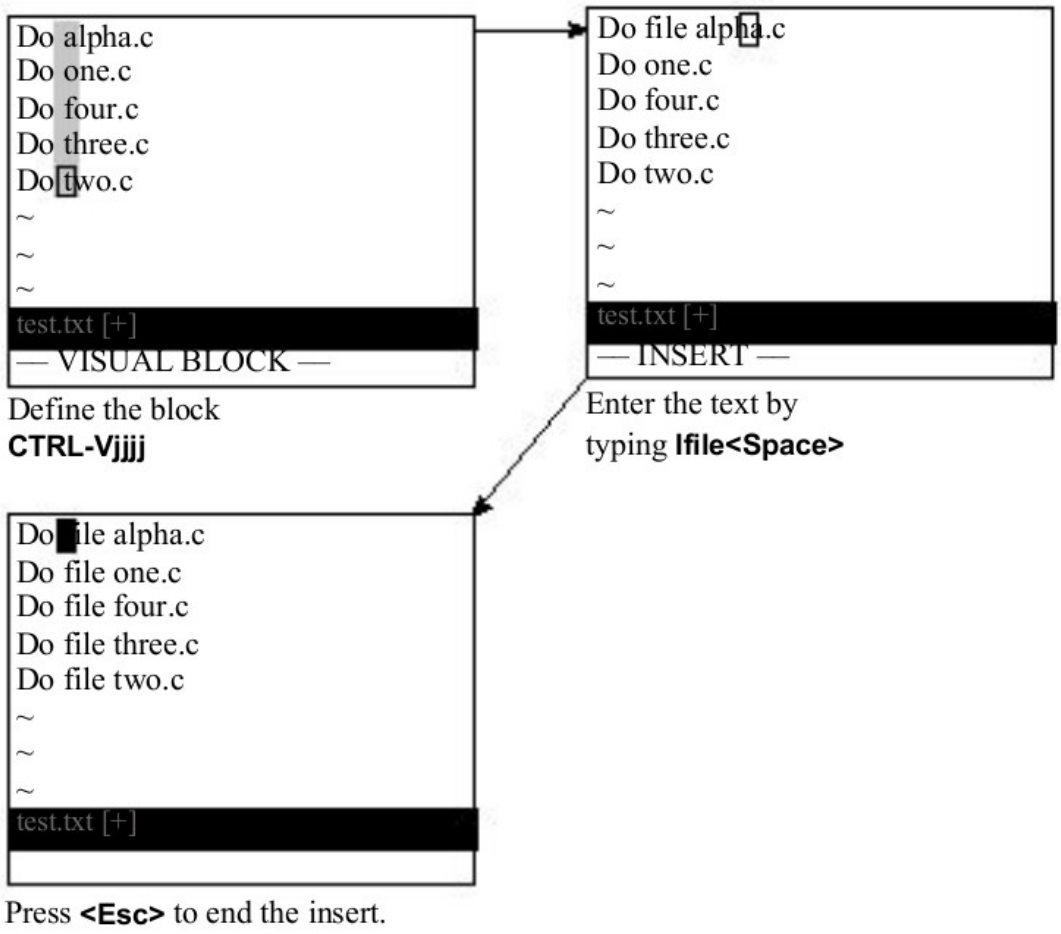
دستور k به دنبال کلمه‌ای که کرسر رویش است در man سیستم می‌گردد
در ویژوال مد متن علامت دار را به عنوان ورودی به دستور man می‌دهد

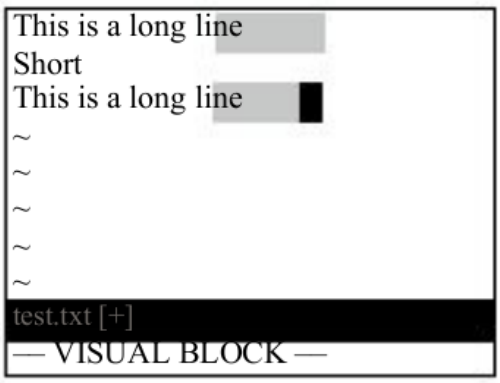
Visual Block Mode

بعضی از دستورات در این وضعیت کمی متفاوت عمل می‌کنند
این وضعیت با CTRL+V شروع می‌شود و می‌توانید مستطیلی متن را انتخاب کند.

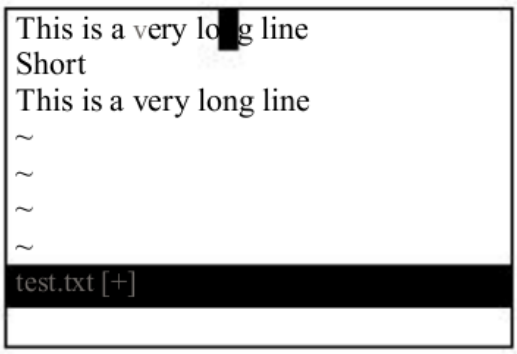
وارد کردن متن

قسمت موردن نظر را علامت بزنید و با I (بزرگ) به اینسرت مد بروید – با این دستور کرسر در ابتدای بلوک و اولین خط آن می‌رود- حالا متن مورد نظر را تایپ کنید و <esc> کنید. متن وارد شد به اول همه خطوط بلوک اضاف می‌شود.
اگر از enter حین وارد کردن متن استفاده کنید این دستور مثل i معمولی عمل می‌کند.
اگر بعضی خطوط کوچکتر از آنی باشند که به بلوک شما برسند انتخاب نمی‌شوند پس I روی آنها تاثیری ندارد.





Select a block. Notice that the short line is not part of the selection.



Insert “very” in the line. Notice that “Short” is unchanged.

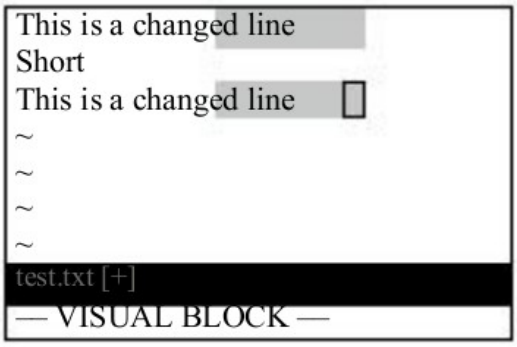
تغییر متن

با دستور c متن علامت زده شده پاک می‌شود و شما وارد اینسرت مد می‌شوید پس از زدن <ESC> متن مورد وارد شده به تمام خطوط علامت زده شده اضافه می‌شود (مثل I)
دستور C از ابتدای بلوک تا آخر خط را پاک می‌کند سپس شما را در اینسرت مد قرار می‌دهد

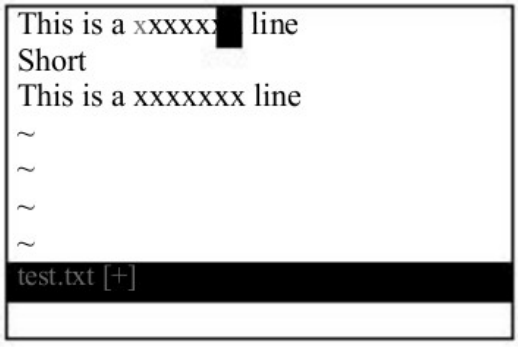
با دستور A می‌توانید متنی به انتهای بلوک اضافه کنید. به خطوطی که به انتهای بلوک نمی‌رسند یا حتی در داخل بلوک قرار ندارند space اضافه می‌شود تا به انتهای بلوک برسند.
اگر حین تعریف بلو ک از $ استفاده کنید تا انتهای همه خطوط انتخاب می‌شوند در این صورت A متن را به انتهای همه خطوط اضافه می‌کند – به آنها space اضافه نمی‌کند-

replacing

با دستور **r**char می‌توانید تمام حروف موجود در بلوک را با حرف داده شده جاجا کنید.
این کار روی خطوط کوچک اثری ندارد.
مثلا rx



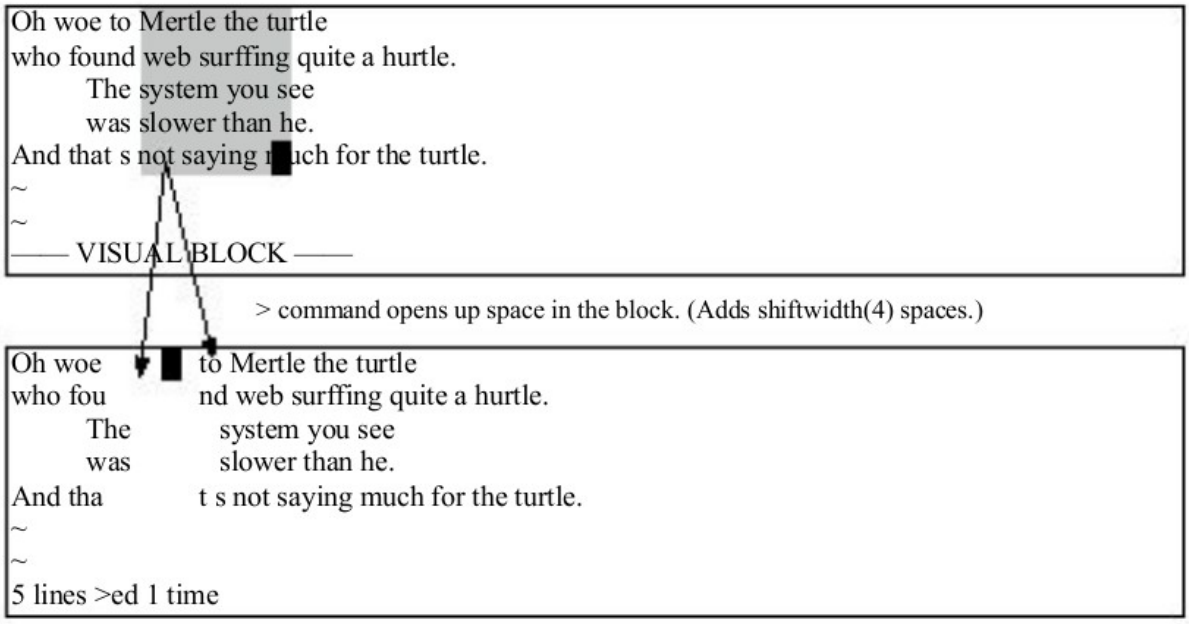
Define the block



The r**x** command replace all characters in the block with "x"

تو رفتگی

دستور > و < مثل قبل عمل می‌کنند فقط مبدا شان ابتدا بلوک می‌شود
یعنی تغییر را نسبت به ابتدای بلوک ایجاد می‌کنند



راهنمایی

برای ویژال بلوک مد از شکل v_b_. استفاد کنید. مثلا برای

:help v_b_r

فرمان‌هایی برای برنامه نویسان

در این فصل یاد می‌گیرید

–رنگ آمیزی سینتکس

–تو رفتگی خودکار

–دستورات مربوط به تو رفتگی

–دستورات حرکن در کد

–گرفتن راهنمای man

–استفاده از تگ

make–

–جستجو در فایل‌ها

رنگ‌آمیزی سینتکس

با این دستور می‌توانید این قابلیت را فعال کنید

:syntax on

یعنی چیزهایی مثل کلمات کلیدی، رشته‌ها، توضیحات و دیگر عناصر سینتکس رنگ خاصی خواهد داشت

شما می‌توانید رنگ هایی که برای این کار استفاده می‌شوند تنظیم کنید

مشکلات

بیشتر مواقع رنگ امیزی سینتکس به خوبی کار می‌کند ولی بعضی مواقع برای تنظیم به کمی مهارت نیاز دارد. اینجا بعضی مشکلات معمول و راه حلشان را می‌بینید

رنگ‌ها بد به نظر می‌رسند

تا به حال سعی کردید که متن زرد روشنی از روی صفحه سفید بخوانید،سخت است! اگر شما چنین چیزی دارید بدانید که مشکل است. ویم دو نوع رنگ امیزی آماده کرده یکی برای زمینه روشن و یکی زمینه تیره. وقتی ویم باز می‌شود چک می‌کند که رنگ زمینه ترمینال شما تیره یا روشن است و سپس متناسب با آن رنگ زمینه را تعیین می‌کند. البته ممکن است که ویم اشتباه کرده باشد.

برای این که ببینید زمینه فعلی شما تیره یا روشن است بزنید

:set backgroun

اگر ویم پاسخ نادرست داد پس شما مقدار را درست کنید

:set background=light

:set background=dark

البته دستور بالا باید قبل از

:syntax on

وارد شود.

من سینتکس را روشن کردم ولی همچنان صفحه سیاه و سفید همیشگی است

مشکل این است که با وجود این که رنگ‌ها برای xterm تعریف شده‌اند ولی ترمینالی که برای xterm تعریف شده رنگ‌ها را از قلم می‌اندازد. این رنگ آمیزی را فلج می‌کند. برای این موضوع ترمینال خو را به نسخه رنگی انتقال دهید.

برای حا این مشکل خطوط زیر را به فایل

$HOME/.cshrc

اضافه کنید

if ($term == xterm) set term = xterm-color

این راه حل روی لینوکس های با شل csh کار می‌دهد. برای دیگر شل‌ها راه حل فرق می‌کند

من یک فایل c ویرایش می‌کند که پسوند استاندارد ندارد. چگونه به ویم بگویم؟

ویم بر اساس پسوند فایل آن را شناسایی می‌کند مثلا فایل‌های c. و h. را به عنوان فایل C درنظر می‌گیرد. ولی اگر هدر شما فایلی به نام settings.inc باشد چه؟ چون این یک فرمت استاندارد نیست ویم نمی‌داند چه کارش کند پس چگونه به ویم بگوییم که این یه فایل C است؟

برای این کار بزنید

:set filetype=c

حالا ویم فایلتان را به عنوان یک فایل C می‌شناسد.

اگر می‌خواهید این تنظیم از این پس خودکار انجام شود این‌جا را ببینید

:help new-filetype

اجرای تست رنگ‌ها

اگر هنوز با رنگ‌ها مشکل دارید می‌توانید با اجرای یک برنامه کوچک تمام رنگ‌ها راببیند و صحت آن‌ها را برسی کنید

این برنامه را با این دو دستور اجرا کنید

:edit $VIMRUNTIME/syntax/colortest.vim

:source %

دستورات شیفت دادن

ویم دستورت زیادی برای ایجاد تورفتگی در کد دارد. اولین دستوری که اینجا برسی می‌شود متن را به چپ << یا راست >> شیفت می‌دهد.

دستور << خط فعلی را به اندازه طول یک شیفت به چپ می‌برد و >>برعکس عمل می‌کند.

ولی طول شیفت چقدر است؟ طول پیشفرض آن ۸ است ولی مطالعات نشان می‌دهد که طول ۴ بهتر است و کد خواناتر است. برای تغییر آن این دستور را بزنید

:set shiftwidth=4

این دستور با عدد همراه می‌شود. مثلا <<5 ۵ خط را به چپ شیفت می‌دهد.

تو رفتگی خودکار

ویم مجموعه‌ای از امکانات تو رفتگی خودکار دارد. شما اصلی ترینشان را اینجا می‌بینید

cindent

این قابلیت برای زبان‌های شبه C کار می‌کند (C, C++, جاوا و ..) و در صورت فعال بودن به طور خودکار بر اساس سبک استاندارد C تو رفتگی ایجاد می‌کند

smartindent

در این صورت برای هر خط به اندازه خط قبلی تو رفتگی ایجاد می‌کند. در صورتی که از } استفاده کنید به طور خودکار یک تورفتگی ایجاد می‌کند و با { آن را بر می‌دارد. برای این که حروف دیگ هم برا تو رفتگی اثر بگذارند cinwords را ببینید

autotindent

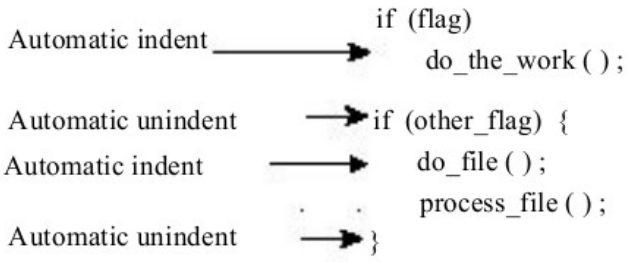
برای خطوط جدید به اندازه خط قبلی تو رفتگی ایجاد می‌شود.

در چند بخش بعدی در مورد این قابلیت‌ها صحبت می‌کنم

Cindent

ویم به خوبی مي داند چگونه برای C یا دیگر زبان‌های ساخت یافته تو رفتگی ایجاد کند. و این کار را به خوبی برای شما انجام می‌دهد فقط کافی است این امکان را فعال کنید

:set cindent



دوست ندارید هر بار که فایلتان را باز می‌کنید تو رفتگی را فعال کنید؟

برای این کار خطوط زیر را به فایل vimrc. اضافه کنید

:filetype on

:autocmd FileType c,cpp :set cindent

خط اول شناسای نوع فایل را در ویم فعال می‌کند و خط دوم اگر فایل C یا C++ بود تو رفتگی را فعال می‌کند.

Smartindent

در این نوع برای هر } یک تو رفتگی ایجاد می‌کند و برای هر { یک تو رفتگی را بر می‌دارد

برای هر عباری که در cinwords تعریف شده باشد یک تو رفتگی ایجاد می‌کند.

نسبت به خطوطی که با # شروع می‌شوند به رفتار خاصی دارد برای این خطوط همه تو رفتگی‌ها را موقتا بر می‌دارد

smartindent کمی از cindent ضعیفتر و از autotindent بهتر است

Autotindent

زبان‌های ساخت یافته‌ای مثل پرل، پاسکال و پایتون از تو رفتگی برای این که بگویند برنامه چه می‌کند استفاده می‌کنند. وقتی به این زبان‌ها برنامه می‌نویسید بیشتر مواقع دوست دارید هر خط به اندازه خط قبلی تو رفتگی داشته باشد.

برای این کار می‌توانید از autotindent استفاده کنید.

مثلا می‌خواهید چنین کدی بزنید

```
if (true) {
    printf("It is true\n");
    printf("It is really true\n");
}
```

مجبورید برای هر دو printf چرا فاصله قرار دهید

ولی اگر این اپشن فعال باشد ویم برای printf دوم چهار فاصه قرار می‌دهد که نتیجه چنین می‌شود

```
if (true) {
    printf("It is true\n");
    printf("It is really true\n");
    }
```

این که { هنوز جلو است خیلی خوب نیست. برای این که { عقب برود بزنید CTRL+D تا خط فعلی یک شیفت به راست رود

دستور =

دستور = روی متن انتخاب شده برنامه داخلی قالب بندی ویم را اجرا می‌کند.

این برنامه با توجه به برنامه شما برایش تو رفتگی ایجاد می‌کند. (شکل)

شما همچنین می‌توانید آن را با یک دستور حرکتی همراه کنید. مثلا دراینجا از % استفاده شده که شما را به براکت جفت براکتی رویش هستید می‌برد.

```
{
if (strcmp (arg, option1) == 0)
return (1);
if (strcmp (arg, option2) == 0)
return (1);
return (0);
}
```

۱- روی اولین } بروید

۲-دستور %= را وارد کنید

```
{
    if (strcmp (arg, option1) == 0)
        return (1);
    if (strcmp (arg, option2) ==0)
        return (1);
    return (0);
}
```

همانطور مه می‌بینید برای کتان تو رفتگی‌های استاندارد ایجاد کرد.

یافتن چیزی در برنامه

ویم برای حرکت در متن دستورات زیادی دارد مثلا

]CTRL+I, [CTRL+I

به دنبال کلمه‌ای گه زیر کرسر است در فایل فعلی و انچه در #include آمده می‌گردد

gd, gD

به دنبال متغیری که زیر کرسر است می‌گردد

]CTRL+D, [CTRL+D

به تعریف ماکرو می‌رود

]d, [d,]D, [D

تعریف ماکرو را نشان می‌دهد

]CTRL-I و I-CTRL[

I-CTRL] به دنبال کلمه‌ای که زیر کرسر است می‌گردد. این کار را از ابتدای فایل فعلی شروع می‌کند و آنچه در include# تعریف شده هم بررسی می‌کند. I-CTRL[همان کار را می‌کند فقط جستجو را از محل فعلی کرسر شروع می‌کند.

gD و gd

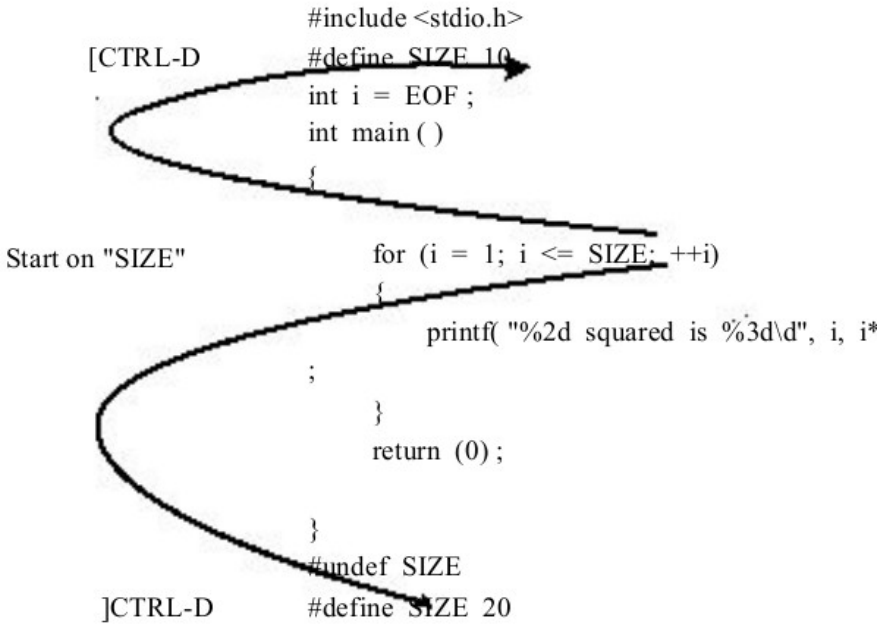
gd به دنبال متغیر محلی ریز کرسر در حوزه خودش می‌گردد. این جستجوی کاملی نیست چون ویم از سینتکس C و C++ اطلاعات کاملی ندارد. بعضی مواقع ممکن است اشتباه کند و متغیرهای عمومی را نشان دهد و یا برعکس ولی در کل خوب عمل می‌کند.

gD هم به دنبال متغیر عمومی زیر کرسر می‌رود. این عهم جستجوی کاملی نیست ولی خوبه!

]CTRL-D و D-CTRL[

این دستور به دنبال تعریف ماکرویی کرسر روآن است می‌رود – این دستور همچنین در فایل‌هایی که include# شده‌اند هم می‌گردد.

] شما را ه اولین تعریف ماکرو می‌رد و [شما را به تعریف بعدی



d] و ...

d] اولین تعریف ماکرویی که اسمش زیر کرسر است نشان می‌دهد

d[همین کار را می‌کند فقط جستجو زا از محل فعلی کرسر شروع می‌کند و اپلین تعریف بعد از کرسر را نشان می‌دهد

در اینجا هم در include# جستجو می‌شود

Results of
[d

```
#include <stdio.h>
#define SIZE 10
int i = EOF;
int main()
{
        for (i = 1; i <= SIZE; ++i)
        {
                printf("%2d squared is %3d\n", i, i*i);
        }
        return (0);
}
#undef SIZE
#define SIZE 20
~
~
~
#define SIZE 10
```

D] همه تعریف‌های ماکرو را نشان می‌دهد

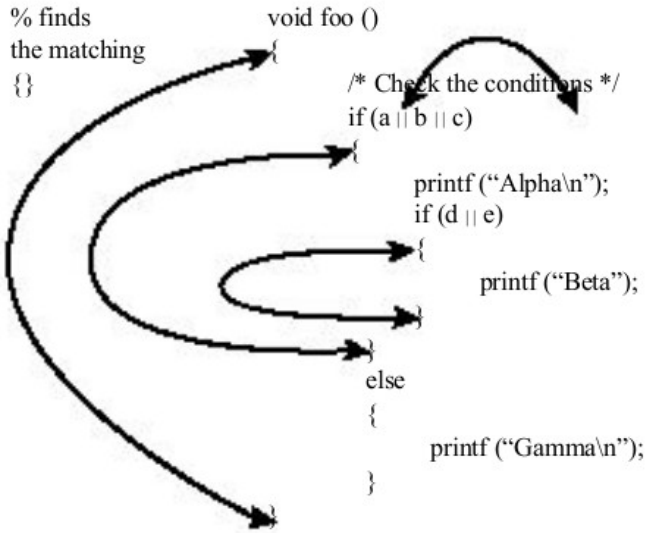
فرق آن با D[در این است که] از اولین تعریف موجو در فایل نشان می‌دهد ولی [از اولین نعریف بعد از محل فعلی کرسر.

Results of
[D

```
int main()
{
        for (i = 1; i <= SIZE; ++i)
        {
                printf("%2d squared is %3d\n", i, i*i);
        }
        return (0);
}
#undef SIZE
#define SIZE 20
~
~
~
test.c
    1:        2   #define SIZE 10
    2:       13   #define SIZE 20
Press RETURN or enter command to continue
```

جفتی‌ها

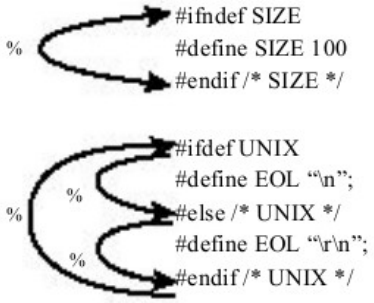
کرسر را روی یکی از جفتی‌ها () [] {} قرار دهید و بزنید % تا رو جفت دیگرش بپرید



این دستور همچینین برای برای توضیحات به سبک C کار می‌کند (محض اطلاع! توضیحات در C با */ شروع و با /* تمام می‌شوند)

همچنین % شما را به endif# متناظر با ifdef# می‌برد (این کار را برای if# و ifndef# هم می‌کند.

برای if# و else# و endif# ٪ شما را از if به else می‌برد و سپس روی endif می‌پرد و دپباره به if بر می‌گردد



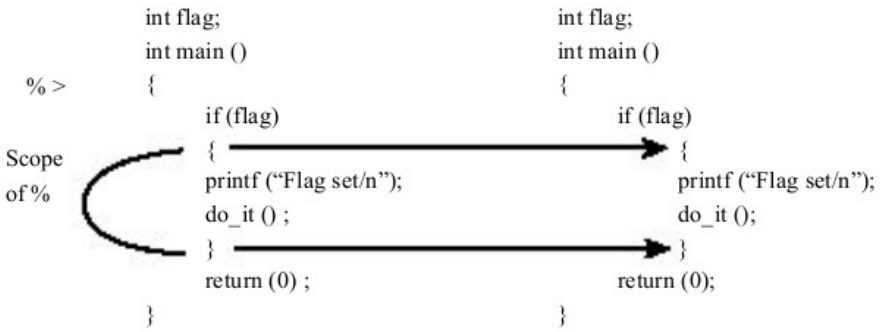
نکته: ویم در مورد جفتی‌ها سیار هوشمندانه عمل می‌کند. ویم در مورد رشته‌ها در سی آگاهی دارد و {} و [ ] داخل رشته‌ها را در نظر نمی‌گیرد

شیفت دادان بللوکی از متن که بین {} است

اگر بخواهید کل متن داخل {} را به اندازه یک شیفت جا به جا کنید

روی یکی از {}های مورد نظر بروید.

بزنید ٪<. این دستور از محل فعلی تا جایی که دستور حرکتی می‌گوید به راست شیفت می‌دهد. دستور حرکتی اینجا شما را به اکلاد جفت می‌برد.

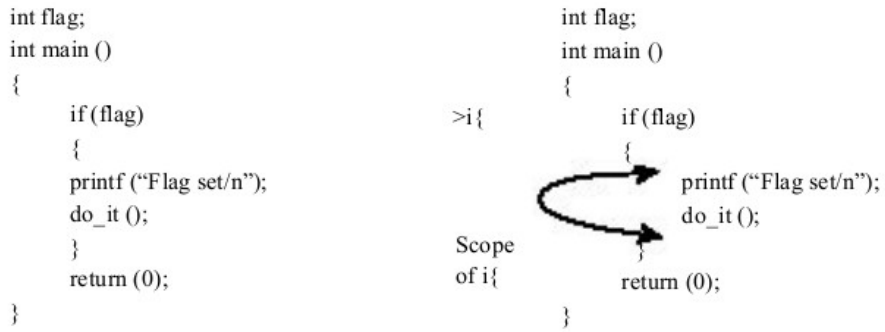


متاسفانه این دستور خود اکلادها را هم حرکت می‌دهد. اگر دوست ندارید اکلادها هم حرکت کنند

۱- روی یکی از اکلادها بروید

۲- بزنید }i<

}i یعنی بلوک داخلیه }[10]



شیفت دادن یک بلوک در ویژوال

۱- روی اکلاد اغازین فرار بگیرید

۲- با v وارد ویژوال شوید

۳- بوسیله {i بلوک داخلی را انتخاب کنید

۴- بوسیله < تو رفتگی ایجاد کنید

یافتن صفحات راهنما

دستور K دستور man یونیکس را رو کلمه‌ای که کرسر رویش است اجرا می‌کند.

در ویندوز K دستور help: را رو کلمه اجرا می‌کند.

دستور man به صورت استنادارد این چنین است

```
$ man [section] subject
```

دستور K ،subject را ارسال می‌کند، ولی اگر بخواهید شما بخش را هم بفرستید چه؟

10 inner {} block

راه حل بسیار ساده است! این دستور عدد می‌گیرد. یعنی اگر شما رو کلمه‌ی mkdir هستید بزنید 2K تا (2)mkdir نشان داده شود

شما می‌توانید تنظیم کنید K چه دستوری را اجرا کند برای این کار 'keywordprg' را باید تغییر دهید

دقت کنید که تعریفی که برای K ایجاد می‌کنید باید از حروفی تشکیل شده باشد که در iskeyword تعریف شده باشند.

تگ

ویم می‌تواند تعریف توابع را در C و C++ پیدا کنید. وقتی می‌خواهید ببنید یک برنامه چگونه کار می‌کند خیلی مفید است.

محل تعریف توابع (که در ویم تگ نامیده شده) در فایل فهرست مانندی که بوسیله برنامه ctags تولید شده ذخیره می‌شود (این برنامه به همراه ویم عرضه شده)

برای این که یک فایل تگ درست کنید از چنین دستوری استفاده کنید

```
$ ctags *.c
```

حالا با این دستور می‌توانید به تعریف تابع مورد نظرتان بپرید

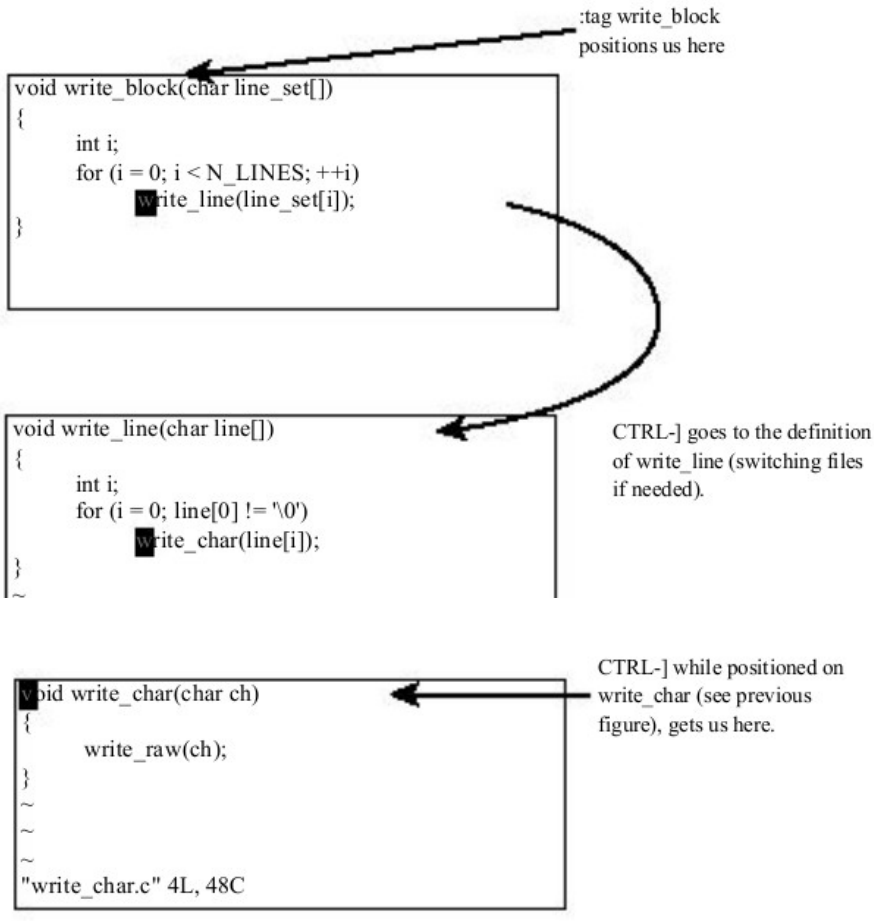
```
:tag function
```

حتی اگر تعریف تابع در فایل دیگری باشد این دستور نشانش می‌دهد

با دستور [+CTRL] می‌توانید به تعریف تابعی که کرسر رویش است بپرید

مثلا شما دارید تعریف تابع write_block را می‌خوانید که این تابع write_line را صدا می‌زند ولی write_line چه کار می‌کند؟ کرسرتان را روی write_line قرار دهید و بزنید [+CTRL]. با این دستور شما به تعریف write_line می‌پرید.

تابع write_line هم چیزی به نام write_char را صدا می‌زند. کرسرتان را روی write_char ببرید و بزنید [+CTRL] تا به تعریف write_char بپرید.



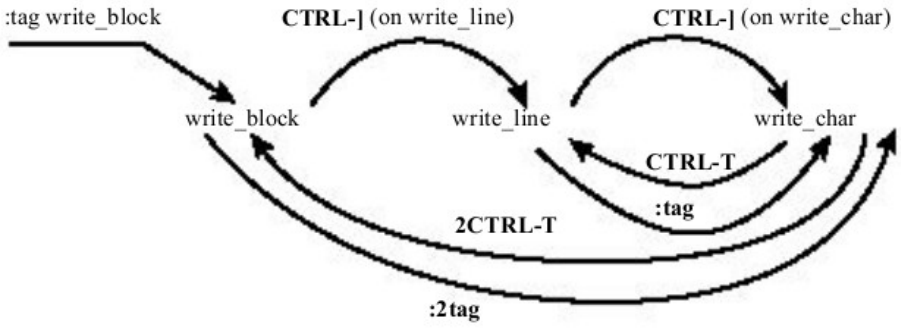
دستور tags: لیست تگ‌های که بینشان حرکت کرده‌اید نشان می‌دهد

```
~
:tags
  #  TO  tag              FROM line    in file/text
  1   1  write_block             1     write_block.c
  2   1  write_line              5     write_block.c
  3   1  write_char              5     write_line.c
>
Press RETURN or enter command to continue
```

اگر می‌خواهید به تگ قبلی برگردید بزنید CTRL+T . این دستور عدد می‌گردد که تعداد تگ‌های که به عقب می‌پرد نشان می‌دهد.

خوب شما جلو رفتید بعد هم عقب رفتید! برای این که دوباره جلو بروید بزنید tag:

این دستور ا عدد هم همراه می‌شود. که نشان می‌دهد چند مرحله به جلو بپرد 3tag:



راهنما و تگ

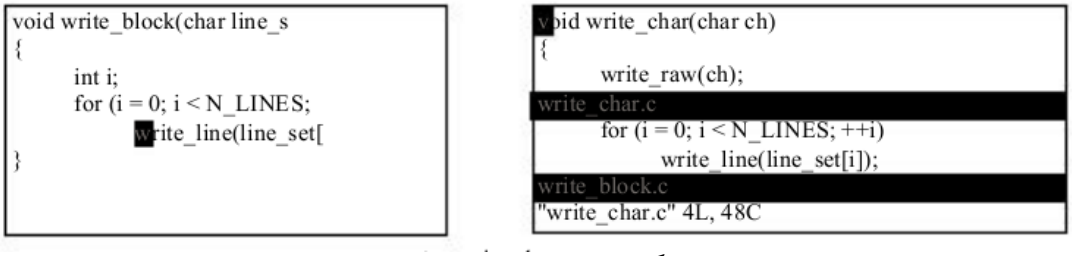
راهنمای ویم از تگ‌ها به صورت سیار گسترده استفاده می‌کند. برای پرش به لینک‌های راهنما از [+CTRL-و CTRL+T-و بقیه دستورات تگ استفاده کنید.

پنجره‌ها و تگ‌ها

وقتی از دستور tag: استفاده می‌کنید پنجره جدید جای پنجره فعلی را می‌گیرد. ولی اگر بخواهید چه؟ شما می‌توانید صفحه را بشکافید پس بزنید.

:stag tag

به تصویر نگاه کنید



دستور [CTRL+W صفحه را می‌شکافد و شما را به تگی که کرسر رویش هستید می‌برد

پیدا کردن یک رویه که قسمتی از آن را به خاطر دارید

ممکن است شما فقط قسمتی از نام تگ مورد نظر را به یاد داشته باشید.

شما می‌توانید از دستور tag: هم برای نمایش یک تابع با نامش استفاده کنید. هم این که به دنبال یک تابع بگردید. برای این کار نام تابع را با / شروع کنید.

مثلا شما به دنبال تابعی به نام something write something می‌گردید. پس بزنید

:tag /write

این دستور نام همه‌ی توابعی که در نامشان write هست نشان می‌دهد

اگر به دنبال توابعی هستید که با read شروع می‌شوند بزنید

:tag /^read

اگر مطمئن نیستید که تابع کدام یک از DoFile و do_file یا Do_File بود بزنید

:tag /DoFile\|do_file\|Do_File

یا

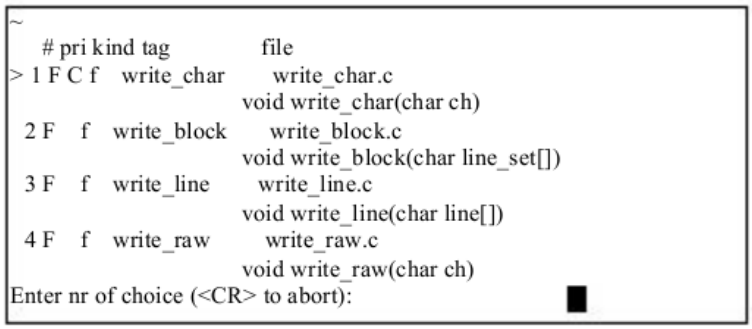
:tag /[Dd]o_\=[Ff]ile

این دستورات تگ‌های مشابه را نشان می‌دهد

شما همچنین می‌توانید برای دیدن لیستی از تگ‌ها از هم می‌توانید

:tselect [name]

استفاده کنید.



Results of :tselect

—ستون اول شماره تگ است

–ستون دوم ستون تقدم است که با این نمادها مشخص شده

–F کاملا منطبق (اگر نبود یعنی در بزرگ و کوچکی حروف فرق دارد)

–S تگ استاتیک (اگر نبود یک تگ عمومی است)

–C تگ در فایل فعلی است

اخرین خط به شما اجازه می‌دهد که با وارد کردن شماره تگ آن را ببینید یا این که برای لغو عملیات فقط Enter کنید

برای این که دستور tselect: رو کلمه‌ای که رویش هستید اجرا شود بزنید [g

دستور tjump: مثل tselect: عمل می‌کند. فقط اگر نتیجه جستجو فقط یک چیز بود به طور خودکار به آن می‌پرد. دستور [+gCTRL دستور tjump: را روی لغتی که زیر کرسر است اجرا می‌کند.

چند دستور دیگر مربوط به تگ هم هست مثلا

:count tnext

به تگ بعدی بپر

:count tprevious

به تگ قبلی بپر

:count tNext

به تگ قبلی بپر

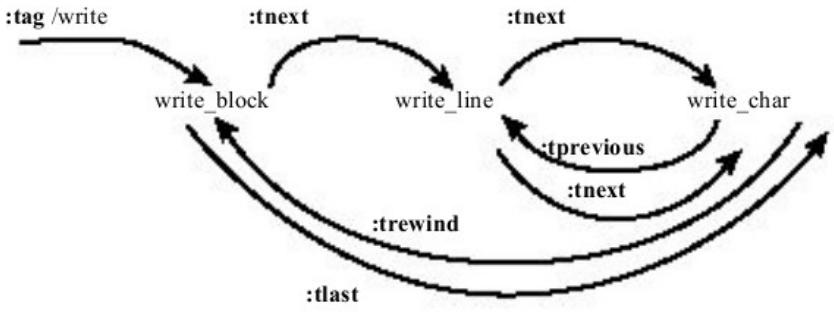
:count tfirst

به اولین تگ بپر

:count tlast

به آخرین تگ بپر

در تصویر نحوه استفاده از تگ‌ها را ببینید



دستور کوتاه

دستوری مثل stselect: دو دستور split: و tselect: را با هم اجرا می‌کنید.

دستور stjump: هم دو دستور split: و tjump: را با هم اجرا می‌کند

نگه‌داری و تغذیه‌ی Makefile

دستور make در یونکیس برای این ساخته شد که برنامه‌ها را کامپایل کند و بسازد. دستورات روط به آن دز فایلی به نام Makefile ذخیره می‌شوند.

فرمت یک Makefile این چنین است، با این که خیلی ساده است ولی برای کار کردن نیاز به کمی مهارت دارد. در اینجا به این که دستور دوم خطا دارد دستور اول درست کار می‌کند.

```
alpha.o: alpha.c
    gcc -c alpha.c
beta.o: beta.c
    gcc -c beta.c
```

شاید پیدا کردن خطا برایتان سخت باشد. دستور اول به اندازه یک tab تو رفتگی دارد ولی دستور دوم به اندازه ۸ space. فهمیدن این خطا روی صفحه خیلی سخت است. پس چونه فرقشان را بفهمیم؟

برای این مشکل ویم به شما وضعیت لیست را عرضه کرده.

بروای ورود به وضعیت لیست بزنید

```
:set list
```

در این وضعیت tab با I^ نشان داده می‌شوند. همچینی بزای نشان دادن پایان خطوط از $ استفاده شده پس می‌توانید پایان خطوط را پیدا کنید.

در صورت استفاده از این دستور مثال ما چنین دیده می‌شود.

```
alpha.o: alpha.c$
^Igcc -c alpha.c$
$
beta.o: beta.c$
    gcc -c beta.c$
```

اینجوری پیدا کردن خطوطی که مشکل دارند راحت‌تر است. (برای دیدن تنظیمات لیست از اپشن listchars استفاده کنید)

اگر اپشن expandtab را فعال کنید وقتی tab بزنید به جای آن ویم Space وارد می‌کند. این برای یک Makefile خوب نیست. برای این که یک tab واقعی وارد کنید مهم نیست گزینه بالا فعال باشد یا نه فقطبزنید CTRL+V<tab>.

CTRL+V به ویم می‌گوید با کارکتر بعدی که می‌زنم کاری نداشته باش

نکته: در صورتی که رنگ‌امیزی سینکتس فعال باشد رنگ خطوطی که با tab شروع شدند رنگشان با خطوطی که با space شروع شدند فرق دارد

مرتب کردن لیستی از فایل‌ها

در Makefileها لیستی از فایل‌ها مثل ایت زیاد می‌بینید

```
SOURCES = \
        time.cpp        \
        set_ix.cpp      \
        rio_io.cpp      \
        arm.cpp \
        app.cpp  \
        amem.cpp        \
        als.cpp         \
        aformat.cpp     \
        adump.cpp       \
        rio.cpp         \
        progress.cpp    \
        add.cpp  \
        acp.cpp  \
        rio_glob.cpp
```

بزای مرتب کردن این‌ها چنین کنید

۱- کرسر را به ابتدای لیست ببرید

۲- آنجا را با دستور ma مارک کنید

۳- به انتهای لیست بروید

۴- دستور خارجی زیر را وارد کنید

```
!'a sort
```

```
SOURCES = \
        acp.cpp     \
        add.cpp     \
        adump.cpp   \
        aformat.cpp \
        als.cpp     \
        amem.cpp    \
        app.cpp     \
```

```
arm.cpp      \
progress.cpp \
  rio.cpp    \
  rio_glob.cpp
  rio_io.cpp  \
  set_ix.cpp  \
  time.cpp    \
```

نکته: همه خطوط باید با \ تمام شوند. مرتب کردن ممکن است این الزام را خراب کند. پس حواستان باشد

مرتب کردن یک لیست در ویژوال مد

۱- به ابتدای متن که باید مرتب شوید بروید

۲- با دستور v وارد ویژوال مد شوید

۳- کرسر را به انتهای متن برسانید

۴- بزنید sort!

! متن انتخاب شده را به دستور pipe می‌کند

Make

شما می‌توانید در ویم برنامه‌تان را کامپایل کنید. خطاها را در ویم ببینید و آن‌ها را رفع کنید و دوباره برنامه‌تان را کامپایل کنید و ...

:make

این دستور make را اجرا می‌کند. شما می‌توانید این دستور را با ارگومان نیز اجرا کنید مثل

:make arguman

اگر خطایی بود نگه داشه می‌شود و ویرایشگر شما را به اولینشان می‌برد

وقتی make کنید ویم خطاهای موجود را نشان می‌دهد (مثل تصویر که نشان می‌دهد در دو فایل main.c و sub.c خطاهایی هست) حالا اگر enter کنید ببینید که ویم شما را به کجا می‌برد (تصویر)

```
:!make |& tee /tmp/vim215953.err
gcc –g –Wall –o prog main.c sub.c
main.c: In function ‘main’:
main.c:6: too many arguments to function ‘do_sub’
main.c: At top level:
main.c:10: parse error before ‘]’
sub.c: In function ‘sub’:
sub.c:3: ‘j’ undeclared (first use in this function)
sub.c:3: (Each undeclared identifier is reported only once
sub.c:3: for each function it appears in.)
sub.c:4: parse error before ‘]’
sub.c:4: warning: control reaches end of non-void function
make: *** [prog] Error 1

2 returned
“main.c” 11L, 111C
(3 of 12): too many arguments to function ‘do_sub’
Press RETURN or enter command to continue
```

اجرای دستور make

```
int main()
{
        int i=3
        do_sub("foo");
          ++i;
          return (0);
}
}

~
(3 of 12):  too many arguments to function  do_sub
```

اولین خطا

بله! ویم شما را به اولین خطا برد و شما مجبور نیستید در فایل‌ها و خطوط حرکت کنید. برای دیدن خطای بعدی بزنید

:cnext

نکته: اگر کاربر ویژوال-سی++ هستیدو می‌خواهید با برنامه مایکروسافت nmake برنامه‌تان را بسازید باید اپشن makeprg را تغییر دهید

برای این‌که به خطای قبلی بروید بزنید :cprevious یا :cNext

برای این‌که اولین یا آخرین خطا راببینید بزنید :clast و :cnfirst

برای این‌که اولین خطای فایل بعدی بیاید بزنید :cnfile

اگر فراموش کردید خطای فعلی چیست بزنید :cc

و برای لیست خطاها بزنید :clist

برای این که چند تا از خطاها را ببینید بزنید

:clist 3,5 لیست خطاهای ۳ تا ۵

:clist ,5 لست خطاهای ۱ تا ۵

:clist 5, لیست خطاهای ۵ تاآخر

دستور clist: خطاها را کوتاه شده نشان می‌دهد برای این که متن خطا را ببینید بزنید

:clist!

اگر از دستور make به طوری استفاده می‌کنید که یک فایل خطا تولید می‌کند می‌توانید از ویم بخواهید از آن استفاده کند

:cfile error-file

اگر نام هیچ فایلی خطایی به ویم ندهید ویم از نام پیشفرض که در اپشن errorfile تعریف شده استفاده می‌کند

و در آخر اگر بزنید

:cquit

شما از ویم خارج می‌شوید. فقط ویم مقدار ۱ به عنوان خطا بر می‌گرداند

This is useful if you are using Vim in an integrated development environment and a

normal exit would cause a recompilation

فایل خطا[11]

اپشن errorfile نام پیشفرض فایل خطا برای دستور clist: است. برای تغییر آن بزنید

:set errorfile=error.list

جستجو برای یک رشته

دستور grep: مثل دستور make: عمل می‌کند. دستور خارجی grep را اجرا و خروجی را نشان می‌دهد. (این دستور در ویندوز کار نمی‌کند – اگر می‌خواهید آن را در ویندوز اجرا کنید به http://www.gnu.org سر بزنید)

مثلا برای این که پیدا کنید در کجا از فلان متغیر استفاده شده بزنید

:grep -w ground_point *.c

w-

یعنی به دنبال کلمه کاملا یکسان بگرد

ground_point

نام متغیری هست که به دنبالش می‌گردیم

*.c

می‌گوید در چه فایل‌هایی به دنبالش بگردد

نکته: دستور grep: اطلاعاتی از سینکتس c ندارد پس ممکن است نتیجه‌ها از رشته‌ها و کامنت‌ها باشد

11 Error file

در این دستورات هم می‌توانید از چیزهایی مثل cnext: و clast: و که برای make: معرفی شده استفاده کنید

چند دستور با حال دیگه

شما می‌توانید ویم رو طوری تنظیم کنید که با هر نوع فایلی رفتار خاصی داشته باشد (در این مورد در فصل ۱۳ سخن خواهیم گفت)

همچنین می‌توانید خطوطی در فایلتان قرار دهید که به ویم بگوید با آن فایل خاص چه رفتاری بکند (در این مورد هم به موقعش صحبت می‌کنیم) :دی!

کوته‌سازی، نگاشت کلیدها و Initialization Files

ویم قابلیت‌هایی دارد که شما را قادر می‌سازد کارهای تکراری را به صورت خودکار انجام دهید.

یکی از آن‌ها کوته‌سازی[12] است که شما بخشی از یک عبارت را تایپ می‌کنید و ویم ادامه آن را وارد می‌کند. شما می‌توانید کلیدهای جدیدی برای دستورات تعریف کنید. و در انتها تنظیمات خود را در یک فایل initialization ذخیره کنید که دفعه ویم هنگام آغاز ه کار ویم به طور خودکار خوانده می‌شود.

کوته‌سازی

مختصر سازی یعنی به کار بردن یک کلمه کوتاه به جای بلند. برای مثال ad به جای advertisement. در ویم می‌توانید یک عبارت مختصر را وارد کنید و خودکار گسترشش دهید. برای این که این یک مختصر سازی را به ویم بفهمانید در دفعه وارد کنید

:abbreviate ad advertisement

حالا هر بار که عبارت ad را وارد کنید کل عبارت advertisement جایگزنش می‌شود.

وارد می‌کنید	می‌بینید
I saw the a	I saw the a
I saw the ad	I saw the ad
I saw the ad<space>	I saw the advertisement<space>

شما می‌توانید یک اختصار برای چند کلمه بسازید مثلا

:abbreviate JB Jack Benny

به عنوان برنامه نویس من اختصارات عجیبی می‌سازم

:abbreviate #b /**********

:abbreviate #e <space>**********/

من این دو را برای کامن‌های جعبه‌ای ساخته‌ام برای شروع می‌زنم b# و در خط آخر می‌زنم e#

چیزی که باید در مورد e# دقت کنید وجود یک فاصله قبل از ستاره است. ویم به فاصله‌های بین عبارت و مختصر توجهی نمی‌کند برای جستن از این مشکل فاصله را برایش هجی می‌کنم <space>

لیست اختصارها

دستور

:abbreviate

اختصاری که ساخته‌اید را لیست می‌کند.

12 Abbreviations

نکته:

دقت کنید تا وقتی که کلید فاصله یا tab را نزده‌اید اخصار گسترده نمی‌شود پس کلمه-addition با advertisementdition قاطی نمی‌شوند.

نگاشت کلیدها

شما می‌توانید مجموعه‌ای از دستورات را به یک کلید بچسبانید.

مثلا شما می‌خاهید دور بعضی از کلمه‌های کروشه گذارید یعنی amount را به {amount} تبدیل کنید. می‌توانید با دستور map تنظیم کنید ویم با کلید F5 این کار را انجام دهید.

:map <F5> i{<Esc>ea}<Esc>

بیاید بررسی کنیم:

F5

کلید عملیاتی است. مثل یک ماشه به محض فشرده شدن کل عملیات را انجام می‌دهد. (ماشه می‌تواند یک کلید یا یک رشته باشد)

i{<ESC>

کارکتر } را وارد می‌کنید

e

کرسر به انتهای کلمه می‌پرد

a{<ESC>

کارکتر { را به انتهای عبارت می‌چسباند.

پس از این که دستور map را اجرا کردید کل عملیاتی که برای اضافه کردن {} انجام می‌شود این است که کرسر را به ابتدای کلمه ببرید و بزنید F5.

نکته:

هنگام map کردن شما می‌توانید برای وارد کردن F5 از کلید مرتبط اسفاده کنید. یا این که کارکترها را وارد کنید (یعنی بزنید < و Fو 5و >) البته به جای <ESC> نمی‌توانید از کلید Esc رو صفحه کلید استفاده کنید چون این کلید می‌گوید درستور فعلی را لغو کن. برای و ارد کردن Esc آن را حرف یه حرف وارد کنید یا این که پشت سر هم بزنید CTRL-V و <ESC>.

CTRL-V به ویم می‌گوید بی‌خیال کارکتر بعدی شو!

اخطار:

اگر کلید جدیدی که تعریف می‌کنید با یکی از دستورات ویم یکی باشد. دستور جدید جانشین می‌شود.

لیست نگاشت‌ها

دستور

:map

(بدون ورودی) نگاشت‌های شما را لیست می‌کند.

حل مشکل کلید Delete

در بیشتر ترمینال‌ها کلید Backspace و Delete درست کار می‌کنند ولی اگر برای شما این کلیدها جابه‌جا عمل می‌کنند وارد کنید

:fixdel

این دستور تعریف‌های داخلی ویم را برای کلیدهای Backspace[13] و Delete[14] را اصلاح می‌کند.

این دستور فقط بر تعریف کلیدهای ویم اثر دارد و بر جدول نگاشت کلیدهای سیستم‌عامل اثری ندارد. کاربران لینوکس برای تغییر جدول نگاشت کلیدها از دستور loadkeys استفاده کنند

The X Window system also has a keyboard mapping table. If you want to change this table, you need to check out the xmodmap command. Check the X Window system documentation for details on how to use this command

تنظیم کلید Backspace

شما می‌توانید تنظیم کنید کلید backspace در وضعیت insert چطور کار کند.

برای مثال دستور زیر اجازه می‌دهد روی تورفتگی‌های خودکار Backspace بزنید

:set backspace=indent

دستور زیر اجازه می‌دهد روی انتهای خطوط Backspace کنید

:set backspace=eol

یعنی اگر شما رو اولین ستون یک سطر باشد با زدن کلید backspace سطر فعلی به سطر قبلی می‌چسبد.

The following command enables you to backspace over the start of an insert:

:set backspace=start

In other words, you can erase more text than you entered during a single insert command.

شما می‌توانید این گزینه‌ها را همزمان داشته باشد. فقط کافیست آن‌ها را با کاما از هم جدا کنید

:set backspace=indent,eol,start

ذخیره تنظیمات

بعد از این که همه تنظیمات خود را اجرا کردید خوب است که آن‌ها را ذخیره کنید.

با دستور mkvimrc می‌توانید همه تنظیمات را ذخیر کنید

:mkvimrc file

file نام فایلی است که تنظیمات درش ذخیر می‌شود.

13 t_kb
14 t_kD

برای خواندن تنظیمات از این دستور استفاده کنید.

:source file

هنگام شروع به کار ویم به دنبال فایل تنظمات می‌گردد اگر یافت آن را اجرا می‌کند.(اولین فایلی که بافت اجرا می‌شود)

فایل‌های initialization چنینند

UNIX

$HOME/.vimrc

$HOME/_vimrc

$HOME/.exrc

$HOME/_exrc

MS-DOS

$HOME/_vimrc

$HOME/.vimrc

$VIM/_vimrc

$VIM/.vimrc

$HOME/_exrc

$HOME/.exrc

$VIM/_exrc

$VIM/.exrc

وقتی ازنسخه GUI اسفاده می‌کنیدفایل‌های دیگری هم خوانده می‌شودو به دنبال فایل gvimrc در همان آدرس فایل vimrc که در لیست آمده می‌گردد. فایل

$VIMRUNTIME/menu.vim

هم خوانده می‌شود.

برای این که لیست فایل‌های initialization را ببیند بزنید

:version

فایلی که تا به حال در موردش حرف نزده‌ایم .exrc است. ویرایشگر Vi از آن استفاده می‌کرده. ویم تنها در صورتی آن را اجرا می‌کند که هیچ کدام از فایل‌های initialization را نیابد. چون که بسایری از دستورات Vim در Vi اجرا نمی‌شوند. پس احتمالا شما تنظیما ت خود را در .vimrc ذخیره می‌کنید

دستور:mkexrc-تنظیمات-را-در-یک-فایل-exrc-ذخیره-می‌کند-ولی-اگر-می‌خواهید-از همه-توانایی‌های-ویم-استفاده کنی-باید-از mkvimrc استفاده کنید.

فایل .vimrc من

```
:syntax on
:autocmd FileType *       set formatoptions=tcql
    \ nocindent comments&
:autocmd FileType c,cpp set formatoptions=croql
    \ cindent comments=sr:/*,mb:*,ex:*/,://
:set autoindent
:set autowrite
:ab #d #define
:ab #i #include
:ab #b /*************************************
:ab #e <Space>*************************************/
:ab #l /*------------------------------------------ */
:ab #j Jack Benny Show
:set shiftwidth=4
:set hlsearch
:set incsearch
:set textwidth=70
```

این فایل با خطی آغاز می‌شود که رنگ‌آمیزی سینکس را روشن می‌کند.

```
:syntax on
```

چیز بعدی دستوراتی است که باید با توجه به فرمت فایل اجرا شوند.

در اینجا tcql نحوه فالب بندی خطوط را نشان می‌دهد. T شکستن خودکار خطوط c شکست خودکار کامنت‌ها q قالب دادن خودکار و I یعنی خطوط بلند را در insert نشکن

من همچنین تو رفتگی خودکار برای c را خاموش کردم nocindent و کامنت را به مقدار پیشفرض برگرداندم

```
:autocmd FileType *   set formatoptions=tcql
    \ nocindent comments&
```

اگر فایل C یا C++ بود این دستورات به طور خودکار اجرا می‌شوند.

It defines some additional format options, namely adding the comment header for new lines (r) and new lines opened with an O command (o). It also turns on C indentation and defines the "comments" option for C- and C++-style comments.

چون این خط بعد از خطی که برای تنظیم همه فایل‌بود قرار دارد برای فایل‌های C/C++ تنظمات جدید جانشین می‌شوند

```
:autocmd FileType c,cpp set formatoptions=croql
    \ cindent comments=sr:/*,mb:*,ex:*/,://
```

تنظیم بعدی روشن کردن تورفتگی خودکار (ایجاد و رفتگی برای خط فعلی به اندازه قبلی) و روشن کردن ذخیره خودکار (یعی هر وقت لازم شد فایل را به صورت خودکار ذخیره کن)

```
:set autoindent
:set autowrite
```

سپس کوته‌سازی‌ها را قرار دادم

```
:ab #d #define
:ab #i #include
:ab #b /*****************************************
:ab #e <Space>*****************************************/
:ab #l /*---------------------------------------------*/
:ab #j Jack Benny Show
```

سپس اندازه تو رفتگی را ۴ کردم که بررسی‌ها نشان می‌دهد برای برنامه نویسان بهترین است.

```
:set shiftwidth=4
```

دو گزینه بعدی جستجو را خیال انگیز می کند!

```
:set hlsearch
:set incsearch
```

وقتی با فایل‌های متنی کار می‌کنم یک صفحه ستونی می‌خواهم

```
:set textwidth=70
```

آشنایی با وضعیت فرمان[15]

ویم براساس یک ویرایشگر قدیمی به نام Vi ساخته شده که Vi هم بر پایه یک ویرایشگر قدیمیتر به نام Ex ساخته شده. Ex زمانی که هنوز ویرایشگرهای screen-oriented ساخته شوند بسیار معمول بوده. Ex برای ترمینال‌های قدیمی که آن زمان مرسوم بوده استفاده می‌شده سخته شده بود.

با این که Ex یک ویرایشگر line oriented است ولی بسیار قوی است. به طوری که اکنون هم استفاده می‌شود. به وجود دستورات شگرف ویم هنوز خوب است بعضی کارها در وضعیت Ex انجام شوند. پس کسانی که ویم را طراحی کردند به شما این اجازه را می‌دهند که همه دستورات ex را دز وضعیت فرمان اجرا کنید.

هر دستوری که با دو نقطه شروع شود به عنوان یک دستور شبه Ex در نظر گرفته می‌شود. در این فصل فرا می‌گیرید دستورات حالت ex چگونه ساخت یافته‌اند و در مورد پر کاربرد ترین دستورات آن بحث می‌کنیم.

ورود به وضعیت فرمان

اگر می‌خواهید یه دستور تک در وضعیت فرمان وارد کنیدفقط کافی است قبل از دستورتان یک دونقطه (:) بزنید. به طور مثال:

```
:set number
```

یک دستور وضعیت فرمان است. بحث درمورد دستورات وضع فرمان وقتی شماره گذاری خطوط فعال باشد مفهوم‌تر است، پس لطفاآن را فعال کنید.

پس از این که دستور وارد شد شما دوباره به حالت معمولی[16] بر می‌گردید.

برای تغییر وضعیت به وضعیت فرمان وارد کنید:

```
:ex
```

البته Q هم همین کار را می‌کند. برای برگشتن به وضعیت معمولی وارد کنید:

```
:visual
```

دستور Print

دستور print: (خلاصه‌اش p: است) خطوط تعیین شده را چاپ می‌کنید. بدون ورودی خط فعلی را نشان می‌دهد.

محدوده

می‌توانید دستور print را با یک محدوده همره کنید. یک محمدوه سادع می‌تواند 1,5 باشد که خط یک تا پنج را نشان می‌دهد. برای نمایش این خطوط به این صورت عمل کنید.

```
:1,5 print
    1 At one university the computer center was
    2 experiencing trouble with a new type of computer
    3 terminal. Seems that the professors loved to
    4 put papers on top of the equipment, covering
```

15 Command mode
16 Normal mode

5 the ventilation holes. Many terminals broke

توجه کنید لازم نیست بین ۵و print فاصله بگذارید. ولی این کار باعث شد مثال بهتر به نظر برسد.

اگر می‌خواهید فقط سطر ۵ چاپ شود:

:5 print

5 the ventilation holes. Many terminals broke

شما می‌توانید از شماره خطوط ویژه استفاده کنید مثلا شماره خط $ یعنی خط اخر پس به این صورت می‌توانید کل فایل راببینید:

:1,$ print

1 At one university the computer center was

...

36 Notice:

37

38 If your computer catches fire, please turn it

39 off and notify computing services.

محدوده % مختصری برای کل فایل ($1,$) است:

:% print

1 At one university the computer center was

...

36 Notice:

37

38 If your computer catches fire, please turn it

39 off and notify computing services.

نقطه یعنی خط فعلی

:. print

39 off and notify computing services.

شما می‌توانیدشماره خطوط را با توجه به محتوایشان تعیین کنید. خط /pattern/ یعنی اولین خط بعد اینجا که محتوی pattern است. با دستور:

:1 print

1 At one university the computer center was

به طخ اول بروید و سپس از اینجا تا خطی که محتوی trouble است را چاپ کنید

:.,/trouble/print

1 At one university the computer center was

2 experiencing trouble with a new type of computer

به طور مشابه ?pattern? نشانگر اولین خطقبل از محل فعلی است. با دستور

:39 print

39 off and notify computing services.

به انتهای فایل بروید وسپ ازآخرین Notice تاآخریت خط را چاپ کنید.

:?Notice:?,39 print

36 Notice:

37

38 If your computer catches fire, please turn it

39 off and notify computing services.

نشانه‌ها[17]

گفته شد در وضعیت نرمال می‌توانید با دستوری مثل ma جایی از متن را به عنوان a نشانه گذارید. اگر خط اول به a نشانه گذاری شده و خط سوم با z در وضعیت فرمان برای چاپ این خطوط بزنید

:'a,'z print

که معادل است با

:1, 3 print

وضعیت دیداری[18]

شما می‌توانید یک دستور فرمان را روی یک منطقه انتخاب شده در وضعیت دیداری اجرا کنید. ابتدا وارد وضعیت دیداری شوید و سپس خطوطی که می‌خواهید انتخاب کنید. سپس وارد وضعیت فرمان شوید و دسوری وارد کنید برای این کار وارد کنید print:. به محض این که شما وارد کردید : ویم به خط پایینی می‌رود این را نشان می‌دهد:.

:'<,'>

نشانه > در ایتدای منطقه انتخاب شده و < در انتهای محل انتخب شده ایجاد شده‌اند.

دستور جانشینی

دستور substitute: به شما اجازه می‌دهدبرای عمل جانشانی را برای محدوده‌ای از خطوط اجرا کنید. شکل کلی این دستور چنین است:

:range substitute /from/to/ flags

(فاصله‌ها برا خوانایی قرار داده شده‌اند)

این دستور به جای کلیه from ها to قرار می‌دهد. مثلا دستور

:% substitute /Professor/Teacher/

یه همه Professor ها را با Teacher جایگزین می‌کند.

17 Marks
18 Visual mode

نکته:

دستور substitute: هیچ وقت به طور کامل وارد نمی‌شود. جایش از نسخه خلاصه شده‌اش استفاده کنید s: هدف از آوردن دستور به طور کامل وضوح بیشتر است.

به طور معمول دستور substitute فقط اولین رخداد در هر خط را جانشین می‌کند. مثلا دستور بالا خط:

Professor Smith criticized Professor Johnson today.

را به:

Teacher Smith criticized Professor Johnson today.

تغییر می‌هد. رای این که همه رخدادها در خط تغییر کنند پرچم g را اضافه کنید.

:% substitute /Professor/Teacher/g

نتیجه:

Teacher Smith criticized Teacher Johnson today.

بفیه پرچم‌ها چنینند p[19] یعنی هر خطی که تغغیر را چاپ کن.

پرچم c[20] برای هر جا به جایی از شما تایید می‌خواهد:

:1,$ substitute /Professor/Teacher/c

برای دستور بالا ویم چنین چیزی چاپ می‌کند.

Professor: You mean it's not supposed to do that?

replace with Teacher (y/n/a/q/^E/^Y)?

حالا شما باید یکی از این جواب‌ها را انتخاب کنید:

y جابه‌جا کن

n جابه‌جا نکن

a از اینجا به بعد همه را جا به کن (دیگر نپرس)

q خارج شو و تغییر دیگری نده

CTRL-E یک خط بالا برو

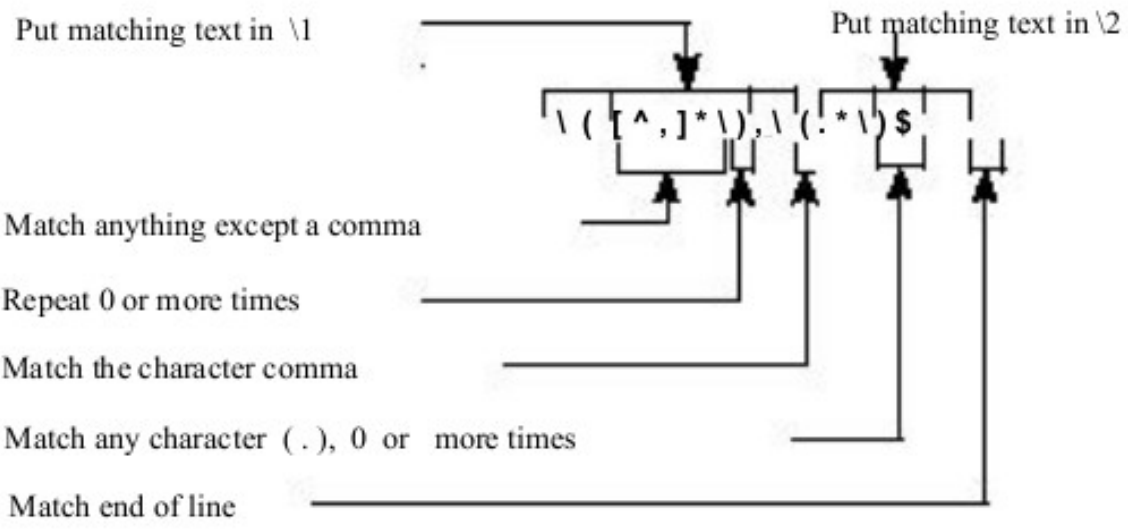
CTRL-Y یک خط پایین برو

جا به جار کردن نام و فامیل

فرض کنید یک فایل محتوی مشخصات به فرمت نام، نام خانوادگی دارید حال می‌خواهید فرمت را به نام خانوادگی، نام تغییر دهید. شما می‌توانید این کار را با یک دستور substitute انجام دهید.

رشته to نام 2\ و فامیل 1\ را می‌گیرد , آن‌ها را پشت سرهم قرار می‌دهد.

19 print
20 Confirm



نوشتن و خواندن فایل

دستور read:[21] یک فایل را می‌خواند و آن را بعد از خط فعلی وارد می‌کند.

دستور write:[22] فایل را ذخیره می‌کند. (این راهی است که کارتان را ذخیره می‌کنید) شما می‌توانید کارتان را در یک فایل جدید ذخیره کنید. برای این کار به دستور write یک آرگمان ورودی به عنوان نام فایل بدهید.

دستور w فایل را جانشین فایل موجود نمی‌کند برای این که ویم را مجبور یا جانشین کردن فایلا کنید write را با یک (!) همراه کنید.

دستور w برای تکه کردن یک فایل بزرگ به بخش کوچکتر بسیار مفید است. فرضض کنید یک فایل با جک‌های زیادی دارید حالا می‌خاهید یکی را برای دوستتان بفرستید. برای این کار در وضعیت دیداری یکی از جک‌ها را انتخاب کنید سپس دستور زیر را وارد کنید:

:'<,'> write joke.txt

دستور shell

دستور shell: شما را به خط فرمان می‌برد. برای خروج از خط فرمان و بازگشت به ویم بزنید exit

:shell

$ date

Mon Jan 17 18:55:45 PST 2000

$ exit

-- vim window appears --

21 :r
22 :w

درباره

علی وکیل زاده

ali.vakilzade@gmail.com

http://isfahanlug.org/doku.php?id=vim