我们知道Linux的目录结构为树状结构，最顶级的目录为根目录 /。

其他目录通过挂载可以将它们添加到树中，通过解除挂载可以移除它们。

在开始本教程前我们需要先知道什么是绝对路径与相对路径。

- 绝对路径
  - 路径的写法，由根目录/写起，例如：`/usr/share/doc`这个目录
- 相对路径
  - 路径的写法，不是由/写起，例如由`/usr/share/doc`要到`/usr/share/man`底下时，可以写成：`cd ../man`这就是相对路径的写法。

文件管理命令

几个常见的处理目录的命令

- ls（英文全拼：list files）: 列出目录及文件名
- cd（英文全拼：change directory）：切换目录
- pwd（英文全拼：print work directory）：显示目前的目录
- mkdir（英文全拼：make directory）：创建一个新的目录
- rmdir（英文全拼：remove directory）：删除一个空的目录
- cp（英文全拼：copy file）: 复制文件或目录
- rm（英文全拼：remove）: 删除文件或目录
- mv（英文全拼：move file）: 移动文件与目录，或修改文件与目录的名称
- touch: 用于创建一个新的文件

touch

创建新文件：用于修改文件或者目录的时间属性，包括存取时间和更改时间。若文件不存在，系统会建立一个新的文件。

```
touch 文件名
```

案例演示

```
# 创建新空白文件
[root@localhost ~]# touch newfile
```

mkdir

创建新目录

```
mkdir [-mpv] 目录名称
```

常用选项

选项	含义
-m	配置文件的权限,直接配置，不需要看默认权限 (umask)
-p	帮助你直接将所需要的目录(包含上一级目录)递归创建起来
-v	显示目录创建的过程

案例演示

```
# 创建一个新的文件夹叫dir
[root@localhost ~]# mkdir dir
# 无法直接创建多层目录
[root@localhost ~]# mkdir a/b/c
mkdir: 无法创建目录"a/b/c": 没有那个文件或目录
# 加上-p选项之后可以自动创建父级目录
[root@localhost ~]# mkdir -p a/b/c
# 显示详细的创建过程
[root@localhost ~]# mkdir -pv dir1/dir2
mkdir: 已创建目录 "dir1"
mkdir: 已创建目录 "dir1/dir2"
```

cp

拷贝文件和目录

```
cp [选项]... 源文件... 目录
```

常用选项

选项	含义
-a	相当于-pdr 的意思，至于 pdr 请参考下列说明
-d	如果源文件是符号链接，则复制链接本身，而不是链接指向的文件
-f	为强制(force)的意思，若目标文件已经存在且无法开启，则移除后再尝试一次
-i	若目标档(destination)已经存在时，在覆盖时会先询问动作的进行
-l	复制符号链接指向的文件，而不是链接本身
-p	连同文件的属性一起复制过去，保持文件的原始属性（如时间戳、权限等）
-r	递归持续复制，用於目录的复制行为
-u	仅在源文件比目标文件新或目标文件不存在时进行复制

选项	含义
-v	显示复制的详细过程

案例演示*

```
[root@localhost ~]# mkdir /home/dir{1,2}
[root@localhost ~]# touch install.log
# 复制文件到目录下
[root@localhost ~]# cp -v install.log /home/dir1
# 复制文件到目录下，并且重命名为abc.txt
[root@localhost ~]# cp -v install.log /home/dir1/abc.txt
# 复制目录
[root@localhost ~]# cp -rv /etc /home/dir1
# 将多个文件复制到同一个目录
[root@localhost ~]# cp -rv /etc/passwd /etc/hostname /home/dir2
# 将多个文件复制到当前目录
[root@localhost ~]# cp -rv /etc/passwd /etc/hostname .
# 备份文件
[root@localhost ~]# cp -rv install.log{,-old}
```

mv

移动文件与目录，或修改名称

```
mv [选项]... 源文件... 目录
```

常用选项

选项	含义
-f	force 强制的意思，如果目标文件已经存在，不会询问而直接覆盖
-i	若目标文件 (destination) 已经存在时，就会询问是否覆盖
-u	若目标文件已经存在，且 source 比较新，才会升级 (update)
-v	显示复制的详细过程

常用实例

```
# 将file1移动到/home/dir2
[root@localhost ~]# touch file1
[root@localhost ~]# mv file1 /home/dir2
# 将file2移动到/home/dir2，并且改名file20
[root@localhost ~]# touch file2
[root@localhost ~]# mv file2 /home/dir2/file20
```

```
# 将file改名为file.txt
[root@localhost ~]# mv file file.txt
```

rm

删除文件或目录

```
rm [选项]... 文件...
```

常用选项

选项	含义
-f	就是 force 的意思，忽略不存在的文件，不会出现警告信息
-i	互动模式，在删除前会询问使用者是否动作
-r	递归删除啊！最常用在目录的删除了！这是非常危险的选项！

案例演示

```
[root@localhost ~]# cd /home
[root@localhost home]# rm -rf dir1
```

禁忌命令

如下命令会从根目录开始删除Linux系统中所有的文件，包括系统文件。严令禁止在任何服务器上输入如下命令

```
# 不要操作！！！
[root@localhost ~]# rm -rf /*
```

文本文件查看

在Linux中一切皆文件，说的就是Linux利用文本文件来保存系统所有的设置。

我们在Linux中想实现一个功能，不可避免的需要查看文本文件，修改文本文件。

cat

用于打开文本文件并显示出来

```
cat [选项]... [文件]...
```

常用选项

选项	含义
-n	由 1 开始对所有输出的行数编号
-b	和 -n 相似，只不过对于空白行不编号
-s	当遇到有连续两行以上的空白行，就代换为一行的空白行
-A	显示控制字符

案例演示*

```
# 查看anaconda-ks.cfg文件
[root@localhost ~]# cat anaconda-ks.cfg
# 查看anaconda-ks.cfg文件，并且显示行号
```

less

可以随意浏览文件，支持翻页和搜索，支持向上翻页和向下翻页

案例演示

```
[root@localhost ~]# less anaconda-ks.cfg
```

head

查看文件的开头部分的内容

```
head [选项]... [文件]...
```

常用选项

选项	含义
-q	隐藏文件名，默认是隐藏
-v	显示文件名
-c N	显示的字节数
-n N	显示的行数

案例演示

```
# 查看文件的前6行
[root@localhost ~]# head -n 6 anaconda-ks.cfg
```

tail

会把文本文件里的最尾部的内容显示在屏幕上

```
tail [选项]... [文件]...
```

常用选项

选项	含义
-f	循环读取
-q	隐藏文件名，默认隐藏
-v	显示文件名
-c N	显示的字节数
-n N	显示文件的尾部 n 行内容
-s	与-f合用,表示在每次反复的间隔休眠S秒

案例演示

```
# 查看文件anaconda-ks.cfg尾部的3行
[root@localhost ~]# tail -n 3 anaconda-ks.cfg
# 查看日志的实时更新情况
[root@localhost ~]# tail -f /var/log/messages
# ctrl+c退出
# 查看文件anaconda-ks.cfg从第10行到结尾
[root@localhost ~]# tail -n +10 anaconda-ks.cfg
```

grep

针对文件内容进行过滤，本工具属于文本三剑客，后续会详细讲解，目前只要求初学者掌握最基本的实例即可

案例演示

```
# 在/etc/passwd的文件中找出有root的行
[root@xwz ~]# grep 'root' /etc/passwd
```

```
# 在/etc/passwd中找出root开头的行
[root@xwz ~]# grep '^root' /etc/passwd
# 在/etc/passwd中找出bash结尾的行
[root@xwz ~]# grep 'bash$' /etc/passwd
```

文本文件编辑

在 Linux 中只掌握文本查看是远远不够的，我们还需要掌握编辑文本文件。

Linux 上也有图形化的文本编辑器，类似于 Windows 的记事本，但是很多时候我们只能用命令行来管理 Linux 操作系统，所以必须要掌握命令行的文本编辑器软件。

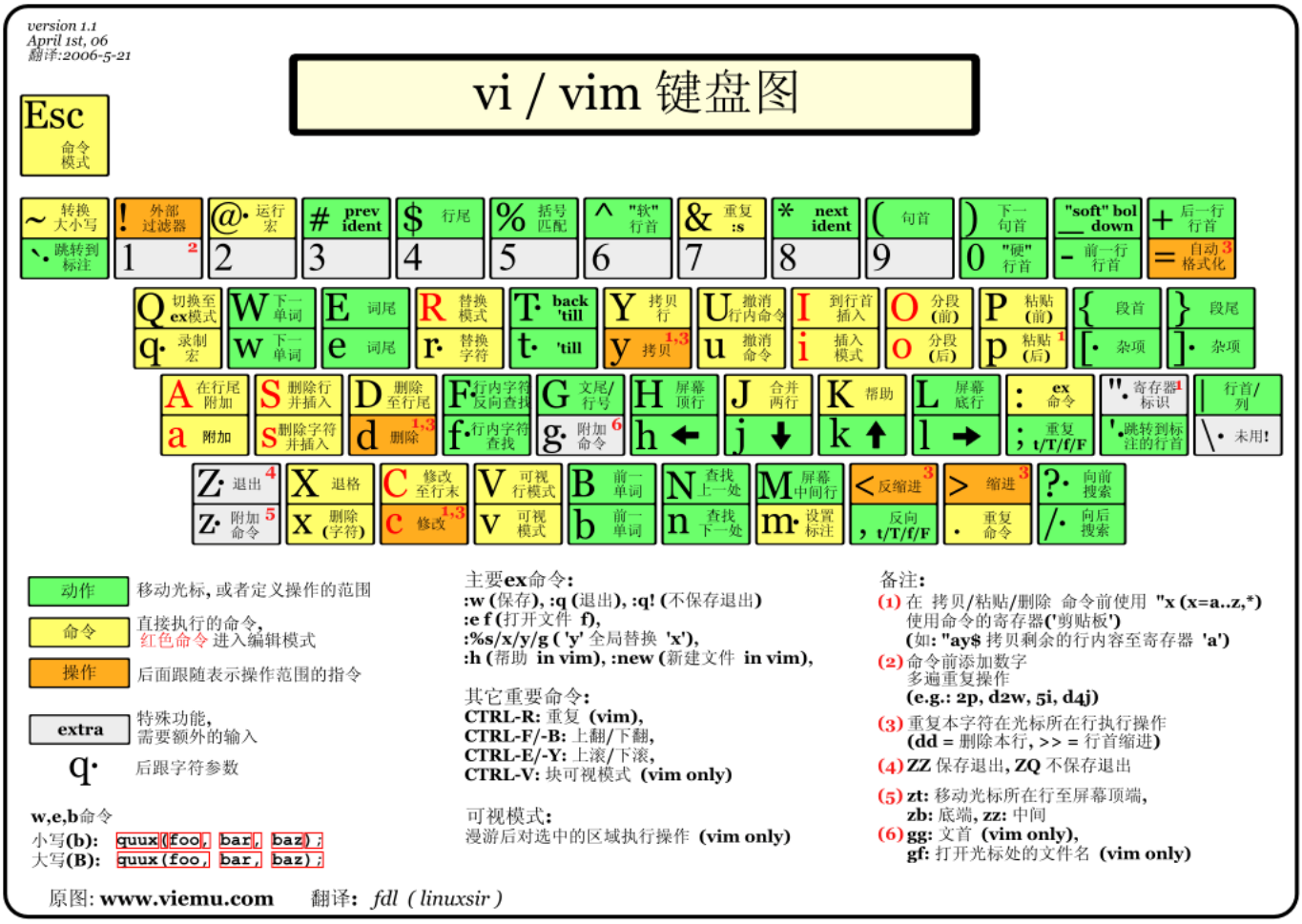
目前常见的命令行文本编辑器

- **nano**：在 Debain 系列的系统上会比较常见，但是其他的 Linux 发行版也都可以安装
- **vi**：所有的 Unix Like 系统都会内建 vi 文本编辑器，其他的文本编辑器则不一定会存在。
- **vim**：具有程序编辑的能力，可以主动的以字体颜色辨别语法的正确性，方便程序设计。

什么是 Vim

Vim 是从 vi 发展出来的一个文本编辑器。代码补完、编译及错误跳转等方便编程的功能特别丰富，在程序员中被广泛使用。

Vim 是一个纯命令行文本编辑器，很多文本编辑的功能都是通过键盘快捷键的方式完成，所以我们需要记住常用的键位。



使用方式

基本上 vi/vim 共分为三种模式，分别是命令模式（Command mode），输入模式（Insert mode）和末行模式（Last line mode）。 这三种模式的作用分别是：

命令模式

用户刚刚启动 vi/vim，便进入了命令模式。

此状态下敲击键盘动作会被Vim识别为命令，而非输入字符。比如我们此时按下`i`，并不会输入一个字符，`i`被当作了一个命令。

以下是常用的几个命令：

命令	含义
`i`	切换到输入模式，以输入字符
`x`	删除当前光标所在处的字符
`:`	切换到末行模式，以在最底一行输入命令

输入模式

在命令模式下按下 `i` 就进入了输入模式。

在输入模式中，可以使用以下按键：

命令	含义
字符按键以及Shift组合	输入字符
ENTER	回车键，换行
BACK SPACE	退格键，删除光标前一个字符
DEL	删除键，删除光标后一个字符
方向键	在文本中移动光标
HOME/END	移动光标到行首/行尾
Page Up/Page Down	上/下翻页
Insert	切换光标为输入/替换模式，光标将变成竖线/下划线
ESC	退出输入模式，切换到命令模式

末行模式

在命令模式下按下 `:`（英文冒号）就进入了末行模式。

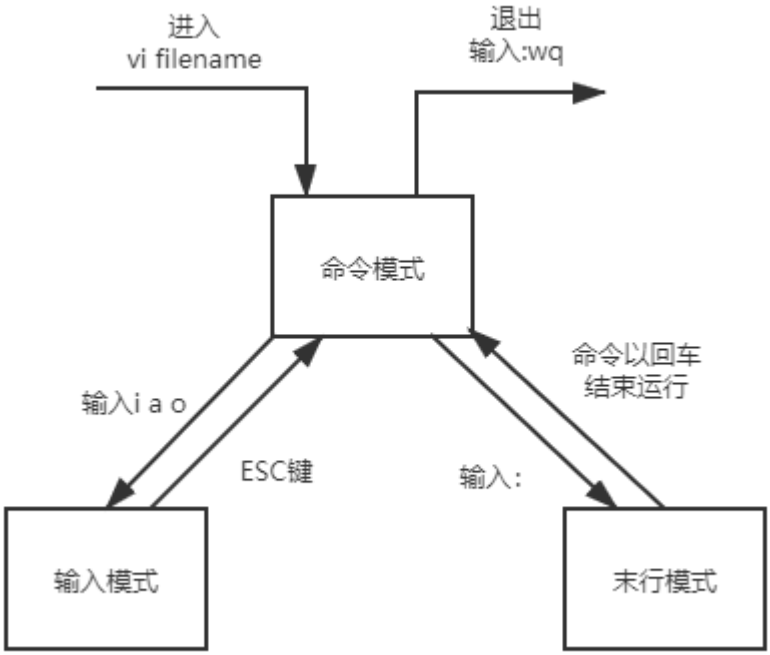
末行模式可以输入单个或多个字符的命令，可用的命令非常多。

在末行模式中，基本的命令有：

命令	含义
q	退出程序
w	保存文件

按 ESC 键可随时退出末行模式。

简单的说，我们可以将这三个模式的关系用下图来表示：



使用实例

有些 linux 发行套件上并没有安装 vim，我们可以安装一下，下面提供了 Rockylinux 的安装命令

```
# yum是在线安装软件的命令，后面会有详细的介绍，这边直接使用
[root@localhost ~]# yum -y install vim
```

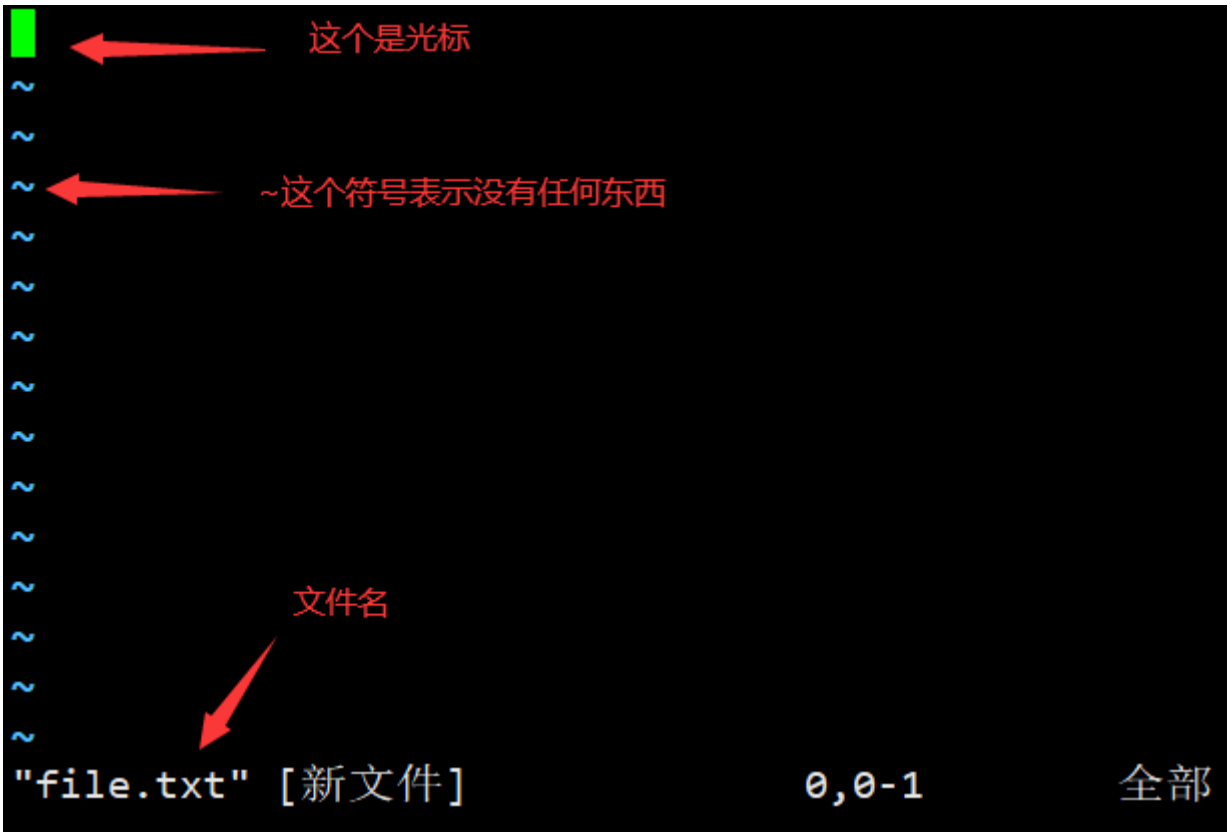
vi和vim大部分的操作完全一模一样，所以会使用vim自然也会使用vi

案例演示

直接输入`vim 文件名`就能够进入 vim 的命令模式了。请注意，记得 vim 后面一定要加文件名，不管该文件存在与否！

```
[root@localhost ~]# vim file.txt
```

输入这条命令之后，会看到如下画面

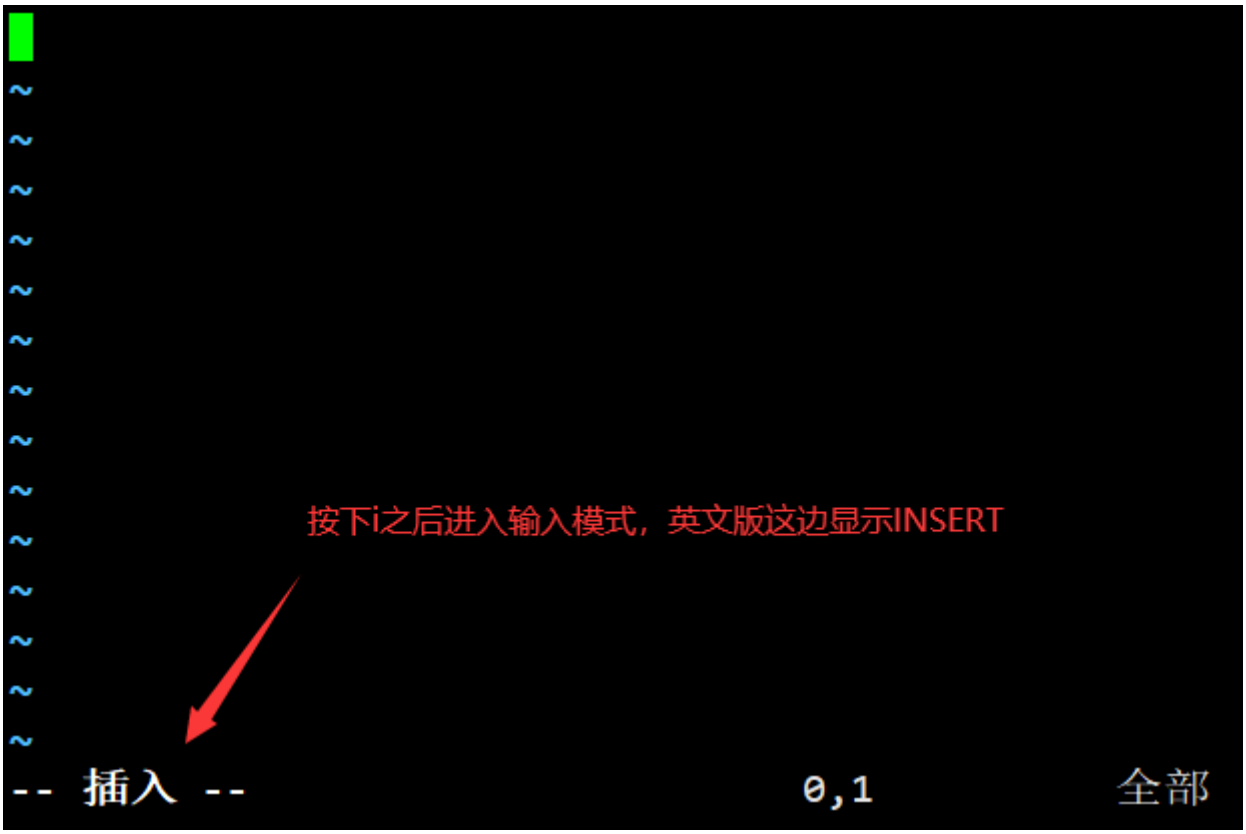


按下`i`进入输入模式(也称为编辑模式)，开始编辑文字

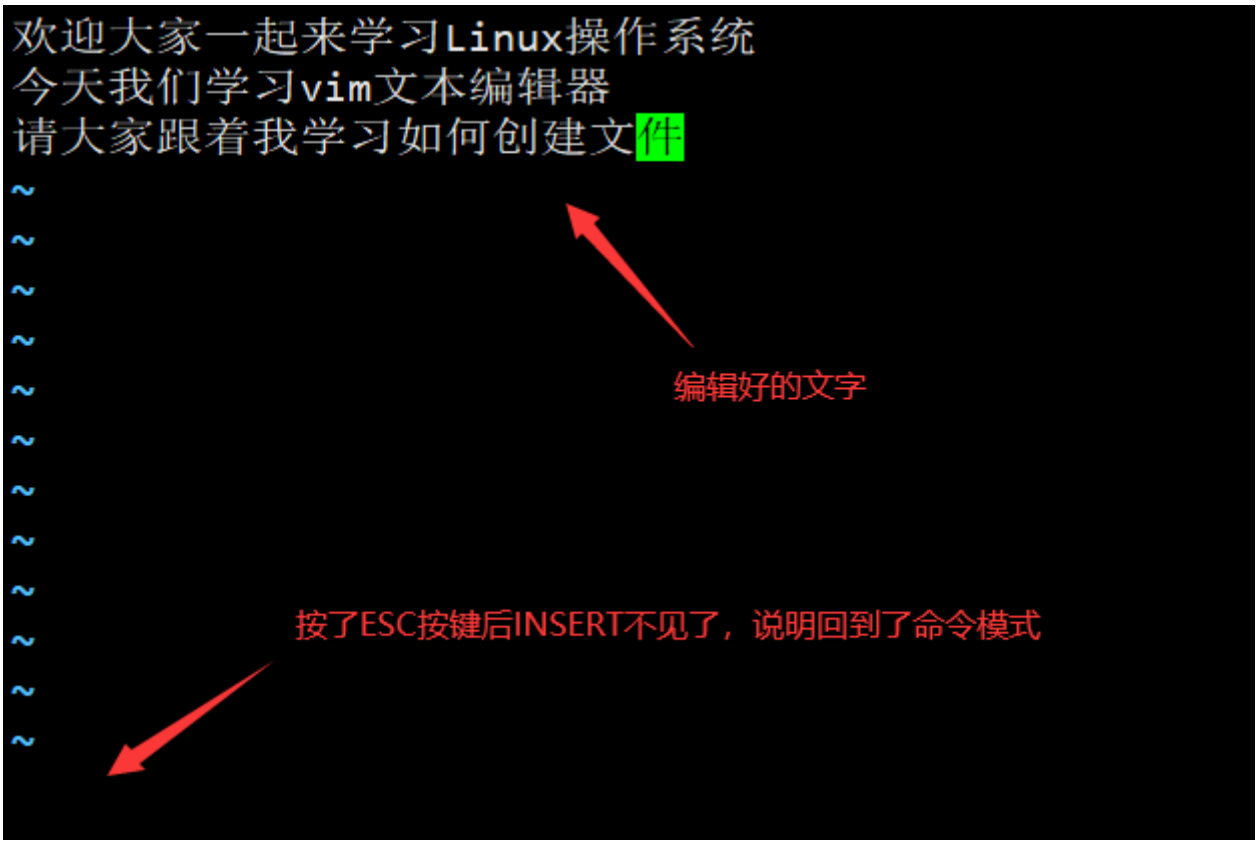
在命令模式之中，只要按下`i`,`o`,`a`等字符就可以进入输入模式了！

在编辑模式当中，你可以发现在左下角状态栏中会出现`--INSERT--`的字样，那就是可以输入任意字符的提示。

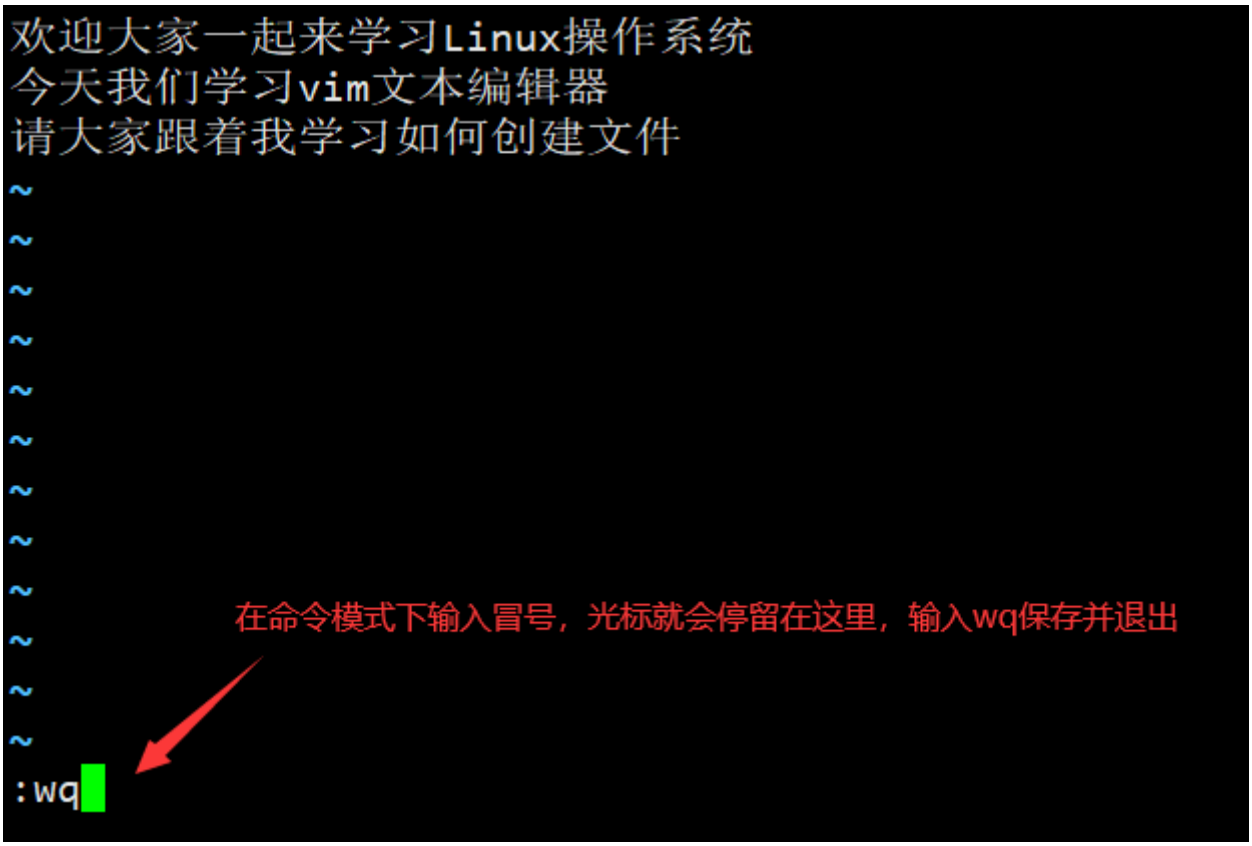
这个时候，键盘上除了 Esc 这个按键之外，其他的按键都可以视作为一般的输入按钮了，所以你可以进行任何的编辑。



好了，假设我已经按照上面的样式给他编辑完毕了，那么应该要如何退出呢？是的！没错！就是给他按下 Esc 这个按钮即可！马上你就会发现画面左下角--INSERT--的不见了！



输入**:wq**后回车即可保存离开，注意其中的冒号必须是英文输入法下的冒号！



现在我们就成功创建了一个文件，查看文件的内容吧

```
[root@localhost ~]# ls -lh file.txt
[root@localhost ~]# cat file.txt
```

按键说明

下面将会列举出 vim 非常多的常用按键，初学者只需要浏览一遍，记住大概 vim 有哪些功能，等后面大量使用 vim 的时候，再来翻阅笔记，并且在多次使用中把这些功能记住。

命令模式

下面的操作都是在命令模式下进行的

移动光标的方法

方法	含义
h 或 向左箭头键(←)	光标向左移动一个字符
j 或 向下箭头键(↓)	光标向下移动一个字符
k 或 向上箭头键(↑)	光标向上移动一个字符

方法	含义
l 或 向右箭头键(→)	光标向右移动一个字符
[Ctrl] + [f]	屏幕『向下』移动一页，相当于 [Page Down]按键 (常用)
[Ctrl] + [b]	屏幕『向上』移动一页，相当于 [Page Up] 按键 (常用)
[Ctrl] + [d]	屏幕『向下』移动半页
[Ctrl] + [u]	屏幕『向上』移动半页
+	光标移动到非空格符的下一行
-	光标移动到非空格符的上一行
n + [space]	那个 n 表示『数字』，例如 20 。按下数字后再按空格键，光标会向右移动这一行的 n 个字符。例如 20 则光标会向后面移动 20 个字符距离。
0 或功能键 [Home]	移动到这一行的最前面字符处
$ 或功能键 [End]	移动到这一行的最后面字符处
H	光标移动到这个屏幕的最上方那一行的第一个字符
M	光标移动到这个屏幕的中央那一行的第一个字符
L	光标移动到这个屏幕的最下方那一行的第一个字符
G	移动到这个档案的最后一行(常用)
nG	n 为数字。移动到这个文档的第 n 行。例如 20G 则会移动到这个档案的第 20 行
gg	移动到这个档案的第一行，相当于 1G
n + [Enter]	n 为数字。光标向下移动 n行

文本的搜索与替换

方法	含义
/word	向光标之下寻找一个名称为 word 的字符串。
?word	向光标之上寻找一个字符串名称为 word 的字符串。
n	这个 n 是英文按键。代表重复前一个搜寻的动作。
N	这个 N 是英文按键。与 n 刚好相反，为『反向』进行前一个搜寻动作。

方法	含义
`:n1,n2s/word1/word2/g`	n1 与 n2 为数字。在第 n1 与 n2 行之间寻找 word1 这个字符串，并将该字符串取代为 word2
`:1,$s/word1/word2/g` 或 `:%s/word1/word2/g`	从第一行到最后一行寻找 word1 字符串，并将该字符串取代为 word2！
`:1,$s/word1/word2/gc` 或 `:%s/word1/word2/gc`	从第一行到最后一行寻找 word1 字符串，并将该字符串取代为 word2！且在取代前显示提示字符给用户确认 (confirm) 是否需要取代！

删除/剪切、复制与粘贴

方法	含义
`x, X`	在一行字当中，x 为向后删除一个字符 (相当于 [del] 按键)， X 为向前删除一个字符(相当于 [backspace] 亦即是退格键)
`nx`	n 为数字，连续向后删除 n 个字符。举例来说，我要连续删除 10 个字符， 『10x』。
`dd`	删除/剪切光标所在的那一整行(常用)
`ndd`	n 为数字。删除/剪切光标所在的向下 n 行，例如 20dd 则是删除 20 行
`d1G`	删除光标所在到第一行的所有数据
`dG`	删除光标所在到最后一行的所有数据
`d$`	删除游标所在处，到该行的最后一个字符
`d0`	那个是数字的 0 ，删除光标所在处，到该行的最前面一个字符
`yy`	复制游标所在的那一行
`nyy`	n 为数字。复制光标所在的向下 n 行，例如 20yy 则是复制 20 行
`y1G`	复制游标所在行到第一行的所有数据
`yG`	复制游标所在行到最后一行的所有数据
`y0`	复制光标所在的那个字符到该行行首的所有数据
`y$`	复制光标所在的那个字符到该行行尾的所有数据
`p, P`	p 为将已复制的数据在光标下一行贴上，P 则为贴在光标上一行！ 举例来说，我目前光标在第 20 行，且已经复制了 10 行数据。则按下 p 后， 那 10 行数据会贴在原本的 20 行之后，亦即由 21 行开始贴。但如果是按下 P 呢？ 那么原本的第 20 行会被推到变成 30 行。
`J`	将光标所在行与下一行的数据结合成同一行
`c`	重复删除多个数据，例如向下删除 10 行，[10c]
`u`	复原前一个动作。
`[Ctrl]+r`	重做上一个动作。
`.`	重复前一个动作。 如果你想要重复删除、重复贴上等等动作，按下小数点『.』就好了！

进入输入或取代的编辑模式

方法	含义
i, I	进入输入模式(Insert mode)： i 为『从目前光标所在处输入』， I 为『在目前所在行的第一个非空格符处开始输入』。
a, A	进入输入模式(Insert mode)： a 为『从目前光标所在的下一个字符处开始输入』， A 为『从光标所在行的最后一个字符处开始输入』。
o, O	进入输入模式(Insert mode)： 这是英文字母 o 的大小写。o 为在目前光标所在的下一行处输入新的一行； O 为在目前光标所在的上一行处输入新的一行！
r, R	进入取代模式(Replace mode)： r 只会取代光标所在的那一个字符一次；R会一直取代光标所在的文字，直到按下 ESC 为止
[Esc]	退出编辑模式，回到一般模式中

末行模式

末行模式下的储存、离开等指令

方法	含义
:w	将编辑的数据写入硬盘中
:w!	若文件属性为『只读』时，强制写入该文件。不过，到底能不能写入， 还是跟你对该文件的权限有关
:q	离开 vim
:q!	若曾修改过文件，又不想储存，使用 ! 为强制离开不储存。
:wq	保存后离开，若为 :wq! 则为强制保存退出
ZZ	这是大写的 Z 喔！如果修改过，保存当前文件，然后退出！效果等同于(保存并退出)
:x	效果等同于(保存并退出)
:X	大写的X，用于加密文件
ZQ	不保存，强制退出。效果等同于 :q!
:w [filename]	将编辑的数据储存成另一个文件（类似文件另存为）
:r [filename]	在编辑的数据中，读入另一个文件的数据。亦即将 『filename』 这个文件内容加到光标所在行后面
:n1,n2 w [filename]	将 n1 到 n2 的内容储存成 filename 这个文件。
:!command	暂时离开 vi 到bash命令行下执行 command 的显示结果！例如 『:! ls /home』即可在 vi 当中察看 /home 底下以 ls 输出的文件信息

vim环境变量修改

方法	含义
`:set nu`	显示行号，设定之后，会在每一行的前缀显示该行的行号
`:set nonu`	与 set nu 相反，为取消行号