编辑器vim（三）和用户管理

主要内容

- vim编辑器：ex命令
- vim编辑器：多文件编辑
- vim编辑器：map命令
- 用户管理

命令行模式

- 正常模式下通过:(执行Ex命令)或 / ?(搜索)等进入命令行模式
- Ex命令支持自动完成，通过Tab以及Ctrl-D选择匹配的命令
- 有许多Ex命令可在最前面指出该命令作用的范围(range)，以行为单位
  - 怎么描述某行?
    - 绝对行号，比如0或者1表示第一行，最后一行可以用$表示
    - .(dot) 表示当前光标所在的行
    - 标记(mark)位置对应的行，如'<和'>表示最近可视模式下选择的内容开始位置和结束位置
    - 搜索某个模式所匹配的行：/{pattern}[/]或?{pattern}[?] 分别表示正向还是反向搜索，注意最后一般用/或?表示模式的结束
    - 上述行标识符之后还可以加上+n或-n，表示最终的行号为原来的行号再加上或减去相应的n，比如.+3表示当前光标所在行之后的第三行
  - 作用的范围可以是某一行，也可是从某一行到另一行之间的多行，起始行和结束行之间以逗号隔开
  - %表示所有行，等价于 1,$

.,.+4	当前行到下面第4行
.,/pattern/-1	当前行到找到pattern的那一行的前一行
.,$	从当前行到文件结尾

命令行模式：编辑命令

命令	描述
:[range]d[elete] [x]	删除指定范围内(缺省当前行)的行(并保存到寄存器x中，缺省为未命名寄存器)
:[range]j[oin]	指定范围的行合并为一行
:[range]y[ank] [x]	将指定范围的行复制到寄存器x(缺省为未命名寄存器)中
:[line]pu[t] [x]	将寄存器中的内容黏贴到指定的行(缺省为光标所在行)的下一行
:[range]co[py] {address} 或:[range]t {address}	将指定范围的行的内容复制到指定的行的下一行
:[range]m[ove] {address}	将指定范围的行的内容移动到指定的行的下一行
:{range}norm[al] {commands}	对于指定范围的行执行后面的正常模式命令，如果不完整，相当于最后附加<ESC>或<C-C>
:[range]s[ubstitute]/{pattern}/{string}/[flags]	对指定范围的行，查找相应的pattern并替换为string
:[range]g[lobal]/{pattern}/[cmd]	对于指定范围的行，如果与pattern给出的模式匹配，则执行后面的Ex命令

:6,9 copy	将第6行到第9行复制到当前行之下
:'<, '> m $	将最近可视模式下选择的行移动到文件结尾
:'<, '> norm A;	将最近可视模式下的各行尾部添加;

命令行模式：替换(:substitute)

:[range]s[ubstitute]/{pattern}/{string}/[flags] 对指定范围的行，查找相应的pattern并替换为string

- 模式,字符串和标志部分缺省/隔开，可以是其他字符，当然不是字母和数字，也不允许是反斜杠、双引号和pipe|，比较常用的是%或+或者?等。
- :s命令中可选的flags主要包括如下选项，如果不指定flags,可以省略最后用于分隔替换字符串和flags的分隔符
  - g(global)表示全局替换所有出现的模式，缺省仅替换每行第一次出现的模式
  - i(ignore)表示忽略大小写 I(don't ignorecase)表示大小写相关
  - c(confirm)表示询问用户是否替换, 提示 replace with XXX (y/n/a/q/l/^E/^Y)? y/n 是否替换，a表示all，现在开始都替换，q表示quit，l表示last，替换这次后结束，Ctrl-E/Ctrl-Y表示向上向下滚动一行
- pattern可以省略，表示最近搜索字符串(:s命令和/?等行间搜索共用搜索模式历史记录)
- string也可以省略，表示删除匹配的内容

:[range]s[ubstitute] [flags]
:[range]& [flags]

- 对于指定范围的行执行最近进行的:s命令，即沿用之前的模式和替换字符串，但可以指定新的标志

:1,. s/word/WORD 从1到当前行中的每行第一个word替换为WORD
:2,10 s/word/WORD/g 第2到10行中的所有word都替换为WORD
:%s/word/WORD/gi 文件中所有大小写无关的word都替换为WORD
:s/word// 表示当前行的第一个word删除
:s%/usrbin%/usr/bin% 当前行的第一个/usrbin替换为/usr/bin

命令行模式：global命令

:[range]g[lobal]/{pattern}/[cmd]

:[range]g[lobal]!/{pattern}/[cmd]

对于指定范围的行，如果与pattern给出的模式匹配，则执行后面的Ex命令cmd

- 缺省为%，即所有行
- 感叹号的版本，即:g!/... 表示如果与pattern给出的模式不匹配，则执行cmd
- 执行的是ex命令,但注意不需要之前的冒号，如果要执行正常模式下的命令，则使用normal命令
- cmd如果省略，相当于p[rint]，即显示匹配的行

:g/^def\s\+ 显示python语言中所有模块级的函数的头部，比如 def f1():
:%g/^\s*#/d 将所有以#开始的行删除，即删除注释行
:g/TODO/copy $ 将所有包含了TODO行复制到文件结尾
:'<,'>g/TODO/normal I# 将最近可视模式下选择的行中包含TODO的行最前面插入#

命令行模式：外部命令

命令	描述
:sh[ell]	启动shell，键入exit返回vim
:!{cmd}	启动shell执行命令cmd, 比如!date
:range !{filter}	启动一个shell来执行命令filter，该命令的输入来自于range所指定的内容，输出替代了原有range部分的内容
:[line]r[ead] {file}	在当前或指定行后插入file中的内容
:[line]r[ead] !{cmd}	启动shell执行命令cmd，该命令的标准输出插入到指定的行(或当前行)之后
:[range]w[rite][!] [>>] file	将缓冲区中指定范围的行保存到file中，>>表示附加。缓冲区本身不变，缺省为所有行
:[range]w[rite] !{cmd}	启动shell执行命令cmd, 该命令的标准输入来自于range所指定的行，缺省所有行

:!bash %	调用bash执行当前编辑的脚本 **% 表示当前编辑的文件名**(h :_%)
:0r template.txt	在文件头部插入template.txt中的内容
:'<,'>w tmp.txt	将可视模式下最近选择的行保存到文件tmp.txt
:r !date	当前行之后插入date命令执行后的输出
:w !sudo tee % >/dev/null	当前用户没有权限写所打开的文件，调用sudo切换成超级用户来保存
:'<,'> !sort	将可视模式下最近选择的行重新排序

主要内容

- vim编辑器： ex命令
- vim编辑器： 多文件编辑
- vim编辑器： map命令
- 用户管理

编辑多个文件：文件、缓冲区、窗口和标签

★文件加载到内存保存到buffer，
每个buffer记录了其对应的文件名

- 在:edit file时记录该文件名
- :write时将buffer更新到文件

★窗口(window)为buffer的视图

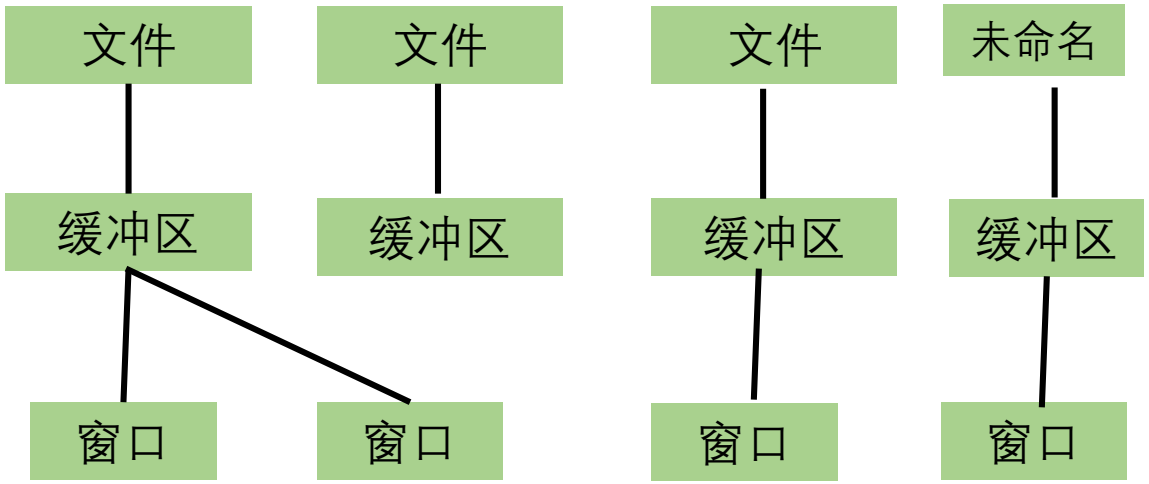
- 一个窗口对应一个缓冲区
- 命令行模式中用%表示当前窗口对应的缓冲区的文件名，用#表示当前窗口最近编辑的替代文件名
- 多个窗口可对应同一个缓冲区，通过一个窗口对于缓冲区的改动会反映到另外一个窗口
- 多个窗口按照纵向、横向等组织呈现给用户
- 缓冲区可以没有窗口对应，称为隐藏缓冲区

★缓冲区列表

- :ls :buffers :files查看当前编辑缓冲区列表
- :ls! 查看所有缓冲区列表，包括:help等打开的特殊缓冲区

★标签(Tab)为组织窗口的方法

- 一个标签页可以包含多个窗口




```
:ls
  1  h    "1.py"                    line 1
  2  h    "demo.py"                 line 1
  3  a +  "format_chinese.py"       line 10
  4 %a    "hello.py"                line 5
  6 #a    "[No Name]"               line 1
```

- 缓冲区列表中包括的字段：
  - 缓冲区ID，可通过 :b[uffer] 4切换到ID为4的缓冲区
  - 状态： u(unlisted)表示仅通过:ls!显示的特殊缓冲区；h(hidden)，没有窗口与其对应; a(active)，有窗口显示该缓冲区。%表示该缓冲区由当前窗口打开, #表示当前窗口最近编辑的缓冲区，称为替代缓冲区. =表示只读，+表示有修改

编辑多个文件：缓冲区

- :edit等命令加载新的文件到当前窗口时原有缓冲区不可见，如果原有缓冲区有改动时命令执行失败，需要通过:edit!等来强制执行，原有缓冲区内容会丢失
- :set hidden，表示:edit等命令切换缓冲区时不考虑原有缓冲区是否有更改，而是隐藏该缓冲区
- :q!或 :qa!命令会直接退出，可能导致隐藏的缓冲区的内容并没有同步到文件中

:ls	列出缓冲区列表, ls!表示包含特殊的缓冲区在内
:e[dit] file|#n	编辑file，其内容加载到当前窗口对应的缓冲区。如果参数为#n，表示编号为n的缓冲区。新缓冲区的文件名对应%，而被切换的缓冲区文件名对应#
:ene[w]	编辑一个新的未命名的文件
:[N]bd[elete] :bd[elete] [N]	卸载当前或指定缓冲区并从缓冲区列表中删除，那些呈现这些缓冲区的窗口也相应关闭。缓冲区有修改时可使用强制版本bd!。N可以是缓冲区编号，也可以是指定范围内的编号，:%bd，相当于1,$bd，即关闭所有缓冲区
:[N]bun[load]	卸载当前缓冲区，但不从缓冲区列表中删除，相应的窗口也关闭
:b[uffer] [N]	编辑编号为N的缓冲区，不给定编号时为仍为当前缓冲区，N也可是缓冲区文件名的一部分，能够与缓冲区列表中的其他文件名区分即可，也可使用自动完成
:bn[ext]	编辑缓冲区列表中的下一个缓冲区
:bN[ext]或: bp[revious]	编辑缓冲区列表中的前一个缓冲区
:bf[irst] :bl[ast] :br[ewind]	编辑缓冲区列表中的第一个、最后一个、第一个缓冲区

编辑多个文件，参数列表

- 在调用vim *.py时表示要编辑多个py文件，这些文件的列表被保存到参数列表(argument list)中，类似地，也可在vim里面调用:args **/*.py命令
- 参数列表与缓冲区列表是两个不同的概念，但又有关联
  - 参数列表记录了多个文件名
  - 调用:args命令更新参数列表时，会自动将参数列表中的文件名加入(但可能未加载)到缓冲区列表中；仅此而已，在删除缓冲区时不会反过来影响参数列表
  - 可通过:next, :previous, :rewind, :first, :last来访问参数列表中的文件名

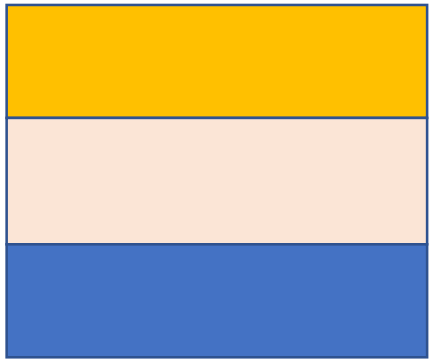
:ar[gs]	打印参数列表的内容，当前参数(文件)以中括号括起来
:ar[gs] {arglist}	更新参数列表， arglist支持通配符扩展， ** 表示各级子目录
:n[ext]	将参数列表中的下一个名字对应的文件加载到当前缓冲区
:N[ext]或:prev[ious]	将参数列表中的前一个名字对应的文件加载到当前缓冲区
:rew[ind]	将参数列表中的第一个名字对应的文件加载到当前缓冲区
:fir[st] :la[st]	将参数列表中的第一或最后名字对应的文件加载到当前缓冲区

编辑多个文件，多个窗口

- vim缺省打开文件编辑时为单窗口方式
- 可通过:split :vsplit等命令水平或垂直创建新的窗口

:hid[e] :clo[se] Ctrl-W c	退出当前窗口（如果与缓冲区为一对一关系，则隐藏该缓冲区），除非是屏幕上的最后一个窗口
Ctrl-W q :q[uit]	退出当前窗口(如果hidden选项设置，且当前缓冲区只有本窗口时，该缓冲区被隐藏)，如果为最后一个窗口时退出vim。
Ctrl-W s :[N]sp[lit] [file]	水平(缺省等分)分割成两个窗口，如果指定file，则新的窗口会打开该文件，否则两个窗口对应同一个缓冲区，缺省新窗口位于上方，可设置:set splitbelow使其位于下方
Ctrl-W v 或:[N]vs[[lit] [file]	与:split类似，只是垂直分割成两个窗口，缺省新窗口位于左边，:set splitright使其位于右方
Ctrl-W n或:[N]new	水平分割并编辑一个新的文件, 等价于 :split | :enew
:[N]vne[w]	垂直分割并编辑一个新的文件, 等价于 :vsplit | :enew
Ctrl-W o或:on[ly]	屏幕上的其他窗口都关闭，剩下当前窗口

Ctrl-W s



Ctrl-W s

Ctrl-W v

编辑多个文件，多个窗口

- 可通过:resize以及 :vertical resize调整窗口大小
- 可通过<Ctrl-W> hjkl切换窗口

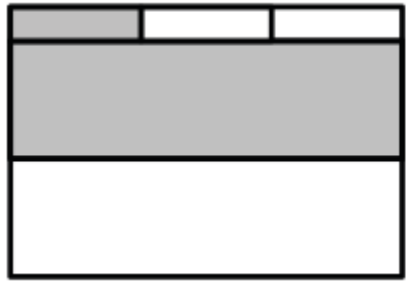
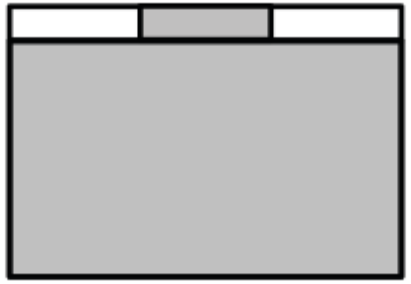
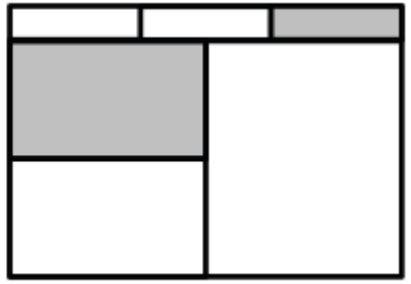
Ctrl-W =	所有窗口的宽度和高度按照相应方向等分
Ctrl-W _ 或:res[ize] Ctrl-W | 或:vertical res[ize]	当前窗口高度以及宽度最大化
:res[ize] {+N|–N|N}	当前窗口高度增加或减少N，或设置为N
vertical res[ize] {+N|–N|N}	当前窗口宽度增加或减少N，或设置为N
Ctrl-W w或Ctrl-W Ctrl-W	在屏幕的窗口之间切换
Ctrl-W h或<Left>	切换到左边的窗口
Ctrl-W l或<Right>	切换到右边的窗口
Ctrl-W j或<Down>	切换到下边的窗口
Ctrl-W k或<Up>	切换到上边的窗口

正常模式下:
C-H <BS> C-J <CR>
C-K nop C-L redraw

```
nnoremap <C-H> <C-W>h
nnoremap <C-J> <C-W>j
nnoremap <C-K> <C-W>k
nnoremap <C-L> <C-W>l
```

命令行模式：编辑多个文件，Tab

- vim的标签页是一个容器，用于容纳一组窗口

- 可以使用:tabedit创建新的标签页
- :tabclose关闭标签页
- :close关闭窗口时，如果其为标签页最后一个窗口，标签页也会关闭

:tabe[dit] {file}	在新的标签页编辑指定的文件或编辑新文件
:tab {cmd}	执行ex命令cmd，如果**该命令打开窗口**时在新的标签页打开 :tab help在新的标签页查看帮助 :tab split 在新的标签页打开当前缓冲区
:tabc[lose]	关闭当前标签页
:tabo[nly]	关闭其他标签页
:tabn[ext] 或gt	切换到下一标签页
:tabp[revious] 或gT	切换到前一标签页
:tabm[ove] [N]	标签页移动到第N个之后，缺省最后一个

定制自己的命令 map/noremap

map [arguments] {lhs} {rhs} 用户在键入lhs中的按键时，相当于执行rhs

- map命令之后为可选的参数，可以出现多次
  - <silent> 表示映射的命令不回显在屏幕上
- vim的map命令支持递归映射，即一个lhs映射为rhs，而rhs可进一步映射
- noremap命令与map命令格式一致，只是其不允许递归映射，建议采用noremap形式
- 可在map和noremap之前加上单字符前缀表示该map仅用于相应的模式
  - map表示正常、可视、选择和操作符等待模式
  - nmap表示正常模式
  - vmap表示可视和选择模式
  - map!表示插入和命令行模式
  - cmap表示命令行模式
  - imap表示插入模式
  - 其他还包括 smap/xmap/lmap等

```
:map j gg
:map Q j
上述map命令执行后，
键入Q等价于键入gg
:noremap Q j  键入Q等
价于j
```

- :map命令可查看映射，:verbose map可查看映射来源于哪里
- 第一个字段表示可用的模式，第二个字段为lhs，接下来第三个字段包含 *表示不能递归映射，第四个字段为rhs

```
n  <C-L>       * :<C-U>nohlsearch<CR><C-L>
n  <C-P>       * :<C-U>CtrlP<CR>
v  <C-S>       * <C-C>:update<CR>
no <C-S>       * :update<CR>
n  ,e            :NERDTreeToggle<CR>
n  ,t            :TagbarToggle<CR>
```

定制自己的命令 map/noremap

- 可通过:map :unmap :mapclear等命令查看、删除和清除映射

:map [arguments] {lhs} {rhs}	将lhs映射为rhs，还可使用nmap/imap/vmap等
:no[remap] [arguments] {lhs} {rhs}	与map类似，只是不允许递归映射
:unm[ap] {lhs}	移除指定模式下lhs映射
:map :nmap等	查看指定模式下的映射列表
:mapc[lear]或:nmapc[lear]	清除指定模式下的映射

- <Leader>表示使用mapleader变量中的按键
  - 缺省为反斜杠\
  - 实践中经常设置为空格或逗号(逗号用于反向重复行内搜索，但使用较少）:let mapleader = ","

```
let mapleader = ","
" 变成超级用户来强制保存文件

" 切换缓冲区   ,l ,h
map <leader>l :bnext<cr>
map <leader>h :bprevious<cr>
```

主要内容

- vim编辑器： ex命令
- vim编辑器： 多文件编辑
- vim编辑器： map命令
- 用户管理

用户管理：用户和用户组

- 用户通过一个唯一的非负的UID（User ID）标识
  - UID=0的用户为超级用户，一般名字为root，拥有系统的所有权限
  - 许多Linux发行版会限制root用户的登录
- 用户组通过一个GID标识
  - 一个用户有一个主用户组，同时也可属于多个从用户组
  - 引入用户组的目的是进行权限管理
- 文件系统的文件和目录可以供哪些用户访问？
  - 其属性部分包括**拥有者ID以及用户组ID**
- 纯粹的数字记忆起来比较麻烦，用户和组除了对应的ID，同样也包括了一个用户名和组名
  - 用户名和组名也应该保证在系统中唯一
  - 用户名可能按照某些规则进行命名，比方说为你的真实姓名中名字的首字母再加上姓（比如我经常用的用户名dlmao）
  - Linux区分大小写，在实践中一般用户名都是采用小写字母

/etc/passwd

```
root:x:0:0:root:/root:/bin/bash
demo:x:1000:1000:demo user,,,:/home/demo:/bin/bash
```

/etc/group

```
/etc/group
root:x:0:
adm:x:4:syslog,demo
sudo:x:27:demo
demo:x:1000:
```

/etc/{passwd,shadow,group}

- 命令行解释器
  - 交互式shell：/etc/shells里面的shell
  - 不允许交互式登录：命令执行的返回值为非0（可以通过echo $?查看），如/usr/sbin/nologin /bin/false等
- 密码：
  - x: 表示密码在shadow文件中
  - 空表示无密码

用户名	密码	用户ID	组ID	描述	主目录	命令行解释器

/etc/passwd

```
root:x:0:0:root:/root:/bin/bash
sshd:x:122:65534::/run/sshd:/usr/sbin/nologin
vboxadd:x:999:1::/var/run/vboxadd:/bin/false
demo:x:1000:1000:demo user,,,:/home/demo:/bin/bash
```

- 密码字段:第一个字符为!或者*时 不允许通过用户名和密码方式登录
  - 超级用户缺省不允许通过密码登录
- 密码并不是明文保存，而是**用户输入的密码进行变换（加密）后的密码（散列值）**

用户名	密码	上次密码更改时间(天)	最小密码age	最大密码age	密码将过期警告间隔	密码不活跃(仍可用)间隔	过期时间(天)	保留

/etc/shadow

```
demo@mars:/etc$ sudo less /etc/shadow
[sudo] password for demo:
root:!:16985:0:99999:7:::
daemon:*:16911:0:99999:7:::
demo:$6$/ayKvXjw$wsP/Vaby1qT981/QOJD/v/J2nWLzagV4isOwJsGSnIeo1
JDlfKuWb4m.oroSIgUFsFC7Puw5Bmyh1nPH.7UDd/:16985:0:99999:7:::
```

组名	密码	组ID	用户列表，以逗号隔开

/etc/group

```
root:x:0:
adm:x:4:syslog,demo
sudo:x:27:demo
demo:x:1000:
```

- /etc/gshadow保存了组密码，在对于用户组进行维护时可以使用该密码
- 每个用户有一个**主用户组**，即/etc/passwd中包含的GID给出的组，同时用户也可以属于多个**从(辅助)用户组**，即/etc/group文件中该用户出现在某些用户组对应的用户列表中

用户管理：查看用户信息

- id命令：id [options] [USER] 查看当前用户或者其他用户的用户和组信息
  - -u 仅显示用户ID
  - -g 仅显示用户组ID
  - -G 仅显示所有用户组ID
  - -n 仅显示名字而不是数字
  - id –un 等价于whoami
- groups命令: groups [USER] 查看当前用户或其他用户所在的组。显示的组中，第一个为主用户组，后面为从组。类似于 id -Gn

```
demo@mars:~$ id
uid=1000(demo) gid=1000(demo) groups=1000(demo), 4(adm),24(cdrom),
27(sudo), 30(dip), 44(video),46(plugdev),113(lpadmin),128(sambashare)
 demo@mars:~$ id root
uid=0(root) gid=0(root) groups=0(root)
demo@mars:~$ groups dlmao
dlmao : dlmao adm cdrom sudo dip plugdev
```

用户管理命令: useradd/groupadd

- 用户管理命令，都要求超级用户权限
- **useradd [options] LOGIN 添加新**用户，设置的主要信息为：用户名、主目录、Shell和用户组

useradd -m -s /bin/bash test1 添加用户test1，创建用户主目录并设置采用bash

  - 如果没有指定，根据缺省配置为用户分配一个唯一的UID、主目录和Shell。注意缺省并不会创建用户主目录
  - 用户的缺省配置文件为/etc/login.defs和/etc/default/useradd，且login.def优先
  - 许多Linux发行版缺省情况下在创建用户的同时也会创建一个相同名字的group，并且为其分配一个组ID
  - 建议采用选项-m(创建用户主目录，并拷贝/etc/skel目录中的文件如.bashrc、.profile等到该目录)和-s SHELL(设置用户所使用的shell)

- passwd test1 为该用户设置一个新的密码

- **groupadd group** 添加新的用户组
- **groupdel group** 删除用户组

```
demo@mars:/etc$ sudo useradd -m -s /bin/bash test1
demo@mars:/etc$ sudo grep test1 /etc/{passwd,shadow,group}
/etc/passwd:test1:x:1002:1002::/home/test1:/bin/bash
/etc/shadow:test1:!:17032:0:99999:7:::
/etc/group:test1:x:1002:
demo@mars:/etc$ sudo passwd test1
[sudo] password for demo:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

用户管理命令: 添加管理员用户

- 用户属于哪些用户组的信息也至关重要
  - 比如sudo组中的用户相当于管理员用户，可以通过sudo获得超级用户权限
  - -g 选项可指定用户的主组(如果不指定，缺省创建一个新的与用户名一样的主组)
  - -G GROUP1[,GROUP2,…[,GROUPN[] 该用户还属于那些用户组

usermod [options] username 修改用户信息

- **-a 将用户加入到其他组(通过-G选项指定)**
- **-G GROUP1[,GROUP2,…[,GROUPN[]** 该用户属于那些组，如果没有-a选项，则从那些不在指定的组里面的组中移走
- -g group 修改用户组
- -l NEW_LOGIN 修改用户名
- -d home 修改用户主目录
- **-L 锁住该用户的口令**
- **-U 解锁用户的口令**

```
demo@mars:/etc$ sudo useradd -m -G sudo -s /bin/bash admin
# 添加一个管理员用户admin，该用户属于sudo组中
demo@mars:/etc$ sudo usermod -a -G sudo test1
```

userdel [options] user 删除用户，缺省情况下不删除用户主目录

- -r 删除用户主目录下的所有文件以及主目录本身

adduser, deluser, addgroup,delgroup等

- 更加高端方便的命令，采用perl脚本语言编写，交互式

adduser [options] user 增加新用户
adduser [options] user group 将用户加入到组
deluser [options] user 删除用户
deluser [options] user group 将用户user从组中移走
addgroup [options] group 增加组
delgroup [options] group 删除组

```
demo@mars:/etc$ sudo adduser demo2
Adding user `demo2' ...
Adding new group `demo2' (1003) ...
Adding new user `demo2' (1004) with group `demo2' ...
Creating home directory `/home/demo2' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for demo2
Enter the new value, or press ENTER for the default
        Full Name []:···
Is the information correct? [Y/n] y
```

```
$ sudo addgroup students
Adding group `students' (GID 1004) ...
Done.
$ sudo adduser demo2 students
Adding user `demo2' to group `students' ...
Adding user demo2 to group students
Done.
```

用户管理：改变自身的信息： passwd/chsh/chfn

- passwd [USER] 改变当前用户或其他用户的密码
- chsh [USER]改变当前用户或其他用户的登录shell
- chfn [USER]改变当前用户或其他用户的全名
- 改变自己的密码/shell/全名并不需要超级用户权限

demo@mars: passwd [options] [LOGIN] 后面不提供用户名时表示对当前用户操作
-d 密码清空，不需要密码可登录
-e 密码过期，用户下次登陆后必须修改密码
-l 锁住密码，无法通过密码登录　　-u 解锁密码

```
demo@mars:/etc$ chsh     #改变shell
Password:
Changing the login shell for demo
Enter the new value, or press ENTER for the default
        Login Shell [/bin/bash]:
```