《十四讲》实践习题 U1-2

笔记本: slam十四讲
创建时间: 2020/3/12 16:04 **更新时间:** 2020/3/12 17:19
作者: 293311923@qq.com
URL: https://github.com/leftthomas/SlamBook/blob/master/ch2/helloSLAM.cpp

helloSlam

第一单元

有线性方程$Ax=b$，若已知A, b，需要求解x，该如何求解？这对A和b有哪些要求？提示：从A的维度和秩角度来分析

有解的条件：b位于$col(A)$的空间内。

求解：**高斯消元法**

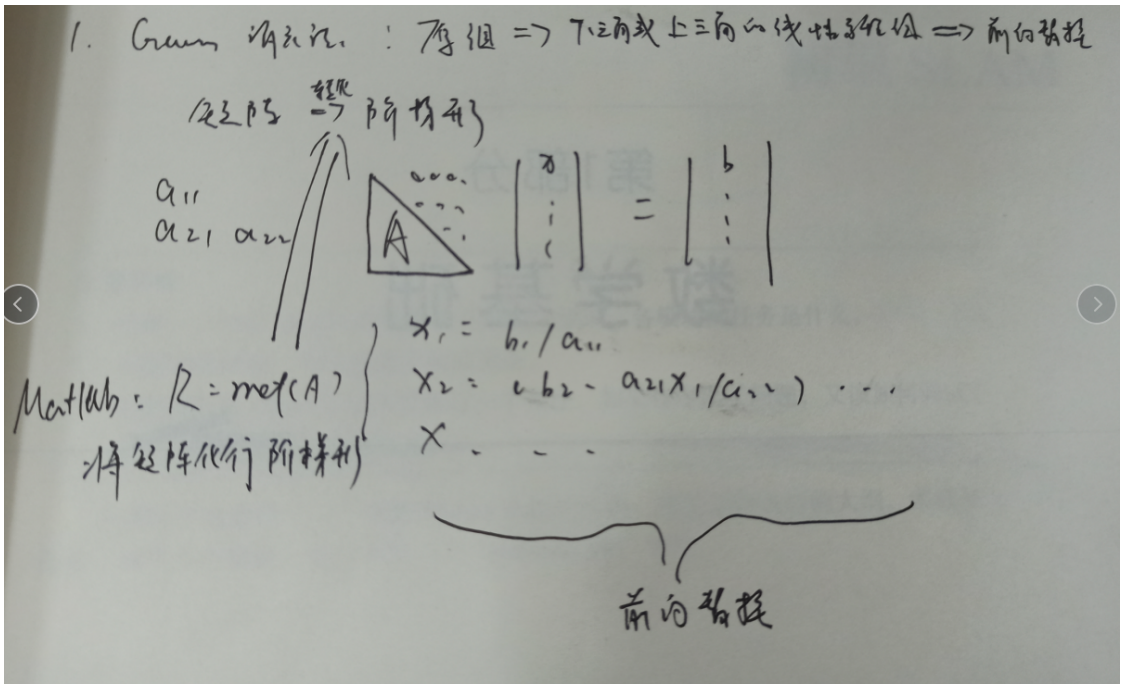
```
def myGauss(m):
    # 行列变换
    for col in range(len(m[0])):
        for row in range(col+1, len(m)):
            r = [(rowValue * (-(m[row][col] / m[col][col]))) for rowValue in
m[col]]
            m[row] = [sum(pair) for pair in zip(m[row], r)]
    # 得到三角矩阵
    ans = []
    m.reverse() # 转置，并后项替换，下面是具体过程
    for sol in range(len(m)):
            if sol == 0:
                ans.append(m[sol][-1] / m[sol][-2])
            else:
                inner = 0
                for x in range(sol):
                    inner += (ans[x]*m[sol][-2-x])
```

```
                # 每次都变换为 ax + b = c 的形式求解
                # 用类似 (c - b) / a的方式求解
                ans.append((m[sol][-1]-inner)/m[sol][-sol-2])
    ans.reverse() #转置并输出结果
    return ans
```

例如我们用一组矩阵进行测试

```
(myGauss([[2.0,4.0,6.0,8.0,10.0,0.0],
                [1.0,3.0,5.0,8.0,3.0,-1.0],
                [3.0,8.0,9.0,20.0,3.0,5.0],
                [4.0,8.0,9.0,-2.0,3.0,8.0],
                [5.0,-3.0,3.0,-2.0,1.0,0]]))
#该表达式得到结果
'''
                        ([1.8835063437139565,
                          1.7012687427912341,
                          -1.4160899653979238,
                          -0.050749711649365627,
                          -0.16695501730103807])
                          '''
```

解出来的结果就对应着未知向量x



高斯分布是什么?它的一维形式是什么样子?它的高维形式是什么样子?

正态分布（英语：normal distribution）又名高斯分布（英语：Gaussian distribution），是一个非常常见的连续概率分布。

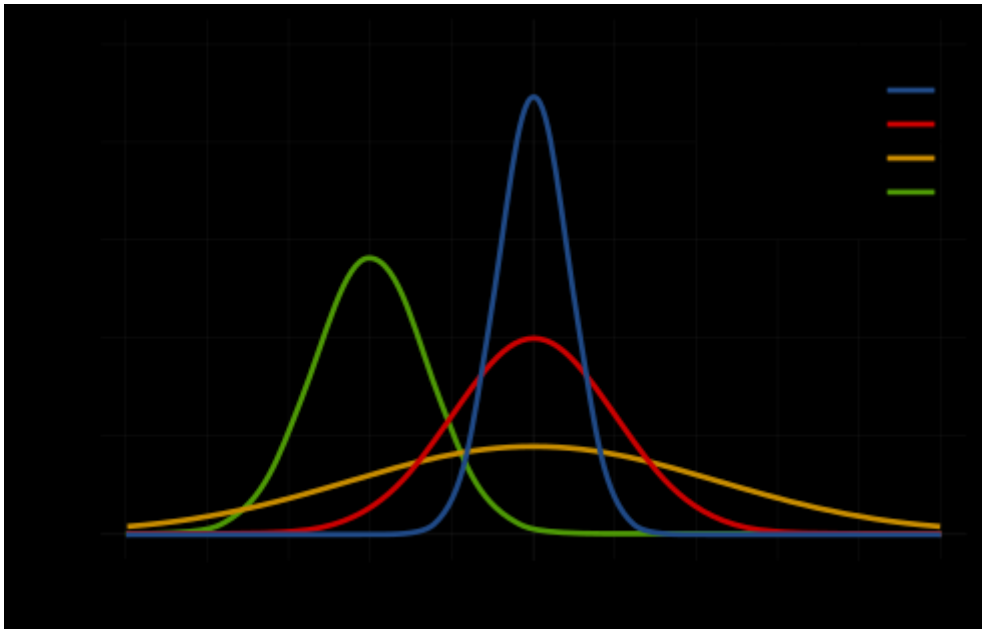
正态分布在统计学上十分重要，经常用在自然和社会科学来代表一个不明的随机变量。

若随机变量 X 服从一个位置参数为μ、尺度参数为 σ的正态分布，记为：$X \sim N(\mu, \sigma^2)$，则其概率密度函数为

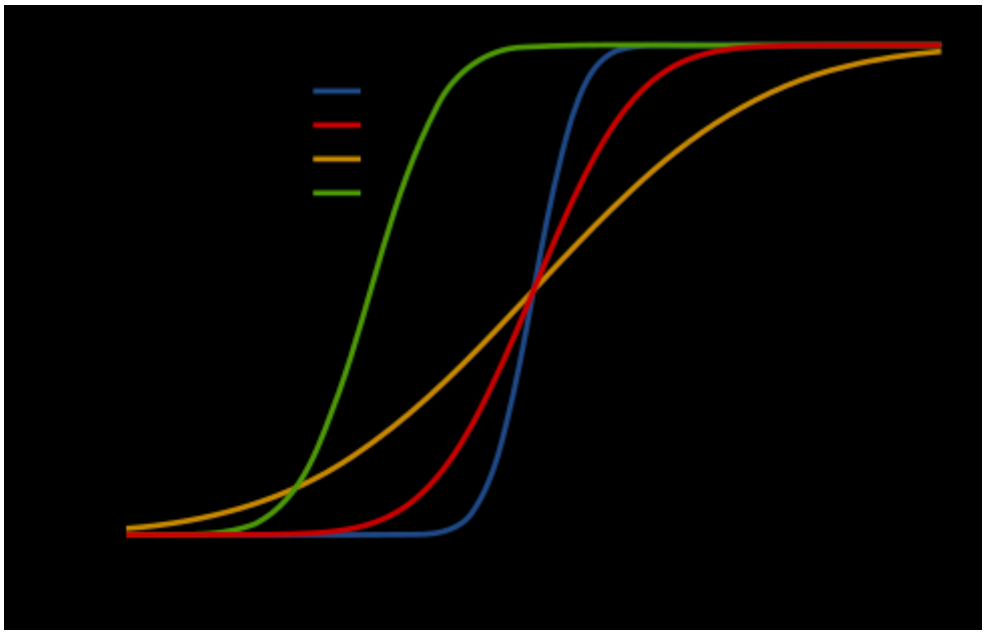
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

正态分布的数学期望值或期望值μ等于位置参数，决定了分布的位置；其方差σ^2的开平方或标准差σ等于尺度参数，决定了分布的幅度。正态分布的概率密度函数曲线呈钟形，因此人们又经常称之为钟形曲线（类似于寺庙里的大钟，因此得名）。我们通常所说的标准正态分布是位置参数$\mu = 0$，尺度参数$\sigma^2 = 1$的正态分布[3]（见下图中红色曲线，红色曲线表示一个标准的高斯分布函数）。

概率密度函数：

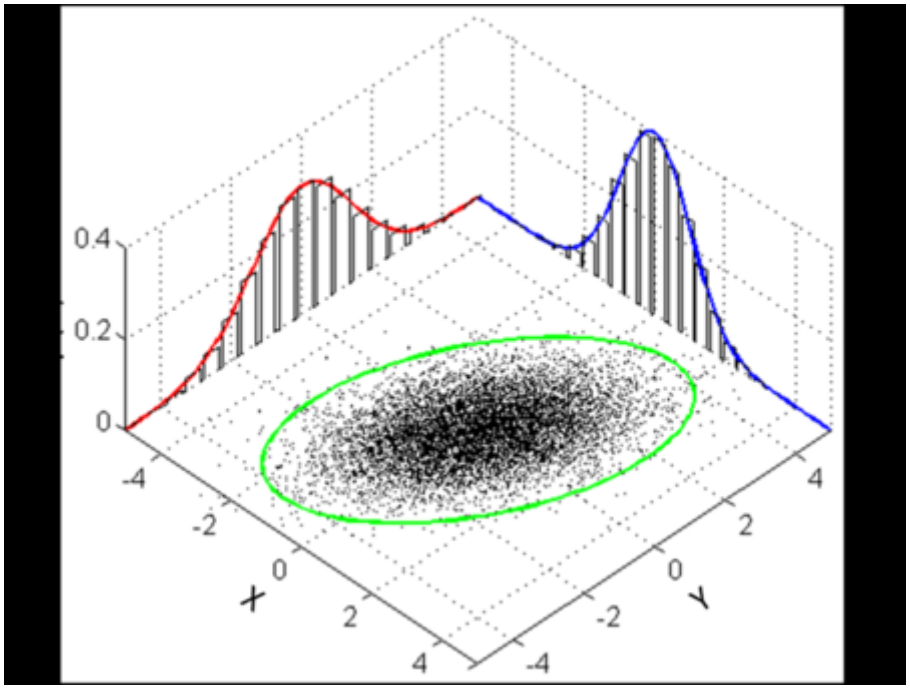


累计分布函数：

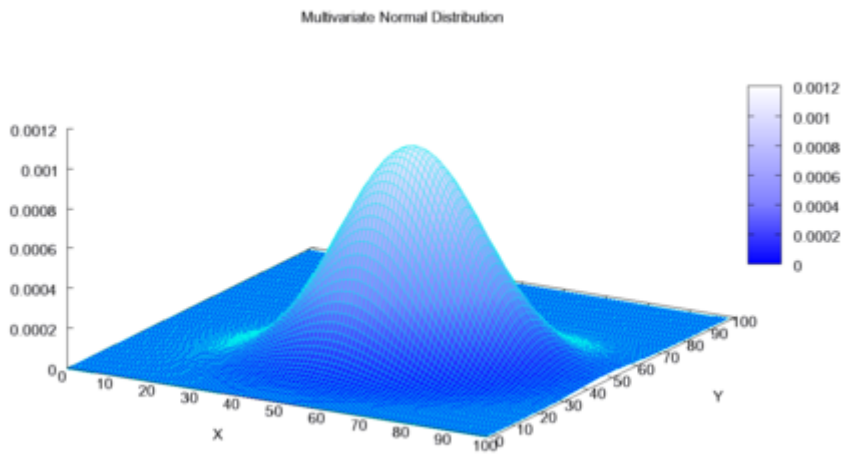


多维高斯分布：

双变量概率密度：



双变量联合概率密度：



你知道 C++ 的类吗?你知道 STL 吗?你使用过它们吗?

STL是Standard Template Library的简称，中文名标准模板库，惠普实验室开发的一系列软件的统称。
它是由Alexander Stepanov、Meng Lee和David R Musser在惠普实验室工作时所开发出来的。

从根本上说，STL是一些“容器”的集合，这些“容器”有list,vector,set,map等，STL也是算法和其他一些组件的集合。这里的“容器”和算法的集合指的是世界上很多聪明人很多年的杰作。

STL的目的是标准化组件，这样就不用重新开发，可以使用现成的组件。STL是C++的一部分，因此不用安装额外的库文件。

STL的版本很多，常见的有HP STL、PJ STL、 SGI STL等。

在C++标准中，STL被组织为下面的13个头文件：**<algorithm>、<deque>、<functional>、<iterator>、<vector>、<list>、<map>、<memory.h>、<numeric>、<queue>、<set>、<stack>和<utility>。**

容器

序列式容器

- 向量(vector) 连续存储的元素<vector>
- 列表(list) 由节点组成的双向链表，每个结点包含着一个元素<list>
- 双端队列(deque) 连续存储的指向不同元素的指针所组成的数组<deque>

适配器容器

- 栈(stack) 后进先出(LIFO)的值的排列 <stack>
- 队列(queue) 先进先出(FIFO)的值的排列 <queue>

- 优先队列(priority_queue) 元素的次序是由作用于所存储的值对上的某种谓词决定的的一种队列 <queue>

关联式容器

- 集合(set) 由节点组成的红黑树，每个节点都包含着一个元素，节点之间以某种作用于元素队的谓词排列，没有两个不同的元素能够拥有相同的次序 <set>
- 多重集合(multiset) 允许存在两个次序相等的元素的集合 <set>
- 映射(map) 由{键，值}对组成的集合，以某种作用于键对上的谓词排列 <map>
- 多重映射(multimap) 允许键对有相等的次序的映射 <map>
  对(pair) 和map类似，但只有一对键值 <utility>
- 智能指针(auto_ptr) 将一个用new开辟内存的指针赋给auto_ptr，会自动回收空间 <memory>

迭代器

模板使得算法独立于存储的数据类型，而迭代器使得算法独立于使用的容器类型。模板提供了存储在容器中的数据类型的通用表示，还需要遍历容器中的值的通用表示，迭代器就是这样的通用表示。

下面要说的迭代器从作用上来说是最基本的部分，软件设计有一个基本原则，所有的问题都可以通过引进一个间接层来简化，这种简化在STL中就是用迭代器来完成的。概括来说，迭代器在STL中用来将算法和容器联系起来，起着一种黏和剂的作用。几乎STL提供的所有算法都是通过迭代器存取元素序列进行工作的，每一个容器都定义了其本身所专有的迭代器，用以存取容器中的元素。

迭代器部分主要由头文件<utility>,<iterator>和<memory>组成。<utility>是一个很小的头文件，它包括了贯穿使用在STL中的几个模板的声明，<iterator>中提供了迭代器使用的许多方法，而对于<memory>的描述则十分的困难，它以不同寻常的方式为容器中的元素分配存储空间，同时也为某些算法执行期间产生的临时对象提供机制,<memory>中的主要部分是模板类allocator，它负责产生所有容器中的默认分配器。

算法

STL提供了大约100个实现算法的模版函数，比如算法for_each将为指定序列中的每一个元素调用指定的函数，stable_sort以你所指定的规则对序列进行稳定性排序等等。这样一来，只要我们熟悉了STL之后，许多代码可以被大大的化简，只需要通过调用一两个算法模板，就可以完成所需要的功能并大大地提升效率。

算法部分主要由头文件<algorithm>，<numeric>和<functional>组成。<algorithm>是所有STL头文件中最大的一个（然而它很好理解），它是由一大堆模版函数组成的，可以认为每个函数在很大程度上都是独立的，其中常用到的功能范围涉及到比较、交换、查找、遍历操作、复制、修改、移除、反转、排序、合并等等。<numeric>体积很小，只包括几个在序列上面进行简单数学运算的模板函数，包括加法和乘法在序列上的一些操作。<functional>中则定义了一些模板类，用以声明函数对象。

你知道 C++11 标准吗?其中哪些新特性你之前听说过或使用过?有没有其他的标准?

革新

- 模板
- decltype关键字，编译期推导表达式类型。
- 可变参数模板（class...、typename... ），是元编程强有力的帮手。

语法糖

- 函数模板的默认模板参数
- using与模板的别名
- auto关键字（自动类型推导）
- lambda表达式（匿名函数对象）
- 初始化参数列表/统一初始化方式

标准库的扩充

- 之前版本中boost库智能指针shared_ptr、unique_ptr扩充为正式的标准库内容。
- 快速组装一个函数对象的bind绑定器。
- 受python等脚本语言启发产生的容器元组（tuple）
- 线程库第一次被纳入到公众库中，从此C++不再需要使用Unix C的API来处理线程和异步。

老语法bug的fixed

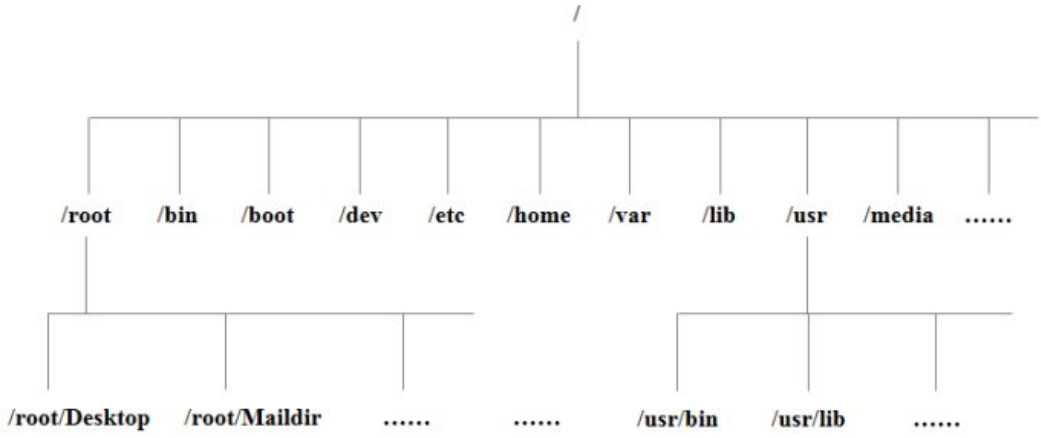
- 之前版本中模板的>>中间不需要加空格了

Linux 的目录结构是什么样的?你知道哪些基本命令

目录结构

- /bin：bin是Binary的缩写, 这个目录存放着**最经常使用的命令。**
- /boot：这里存放的是**启动Linux时使用的一些核心文件**，包括一些连接文件以及镜像文件。
- /dev ： dev是Device(设备)的缩写, 该目录下**存放的是Linux的外部设备**，在Linux中访问设备的方式和访问文件的方式是相同的。
- /etc：这个目录用来存放所有的**系统管理所需要的配置文件和子目录。**

- /home：用户的**主目录**，在Linux中，每个用户都有一个自己的目录，一般该目录名是以用户的账号命名的。
- /lib：这个目录里存放着系统**最基本的动态连接共享库**，其作用类似于Windows里的DLL文件。几乎所有的应用程序都需要用到这些共享库。
- /lost+found：这个目录一般情况下是空的，当**系统非法关机后，这里就存放了一些文件。**
- /media：linux系统会自动识别一些设备，例如U盘、光驱等等，当**识别后，linux会把识别的设备挂载到这个目录下。**
- /mnt：系统提供该目录是为了**让用户临时挂载别的文件系统**的，我们可以将光驱挂载在/mnt/上，然后进入该目录就可以查看光驱里的内容了。
- /opt：这是**给主机额外安装软件所摆放的目录。**比如你安装一个ORACLE数据库则就可以放到这个目录下。默认是空的。
- /proc：这个目录是一个虚拟的目录，它是**系统内存的映射**，我们可以通过直接访问这个目录来获取系统信息。这个目录的内容不在硬盘上而是在内存里，我们也可以直接修改里面的某些文件，比如可以通过下面的命令来屏蔽主机的ping命令，使别人无法ping你的机器：echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all
- /root：该目录为系统管理员，也称作**超级权限者的用户主目录。**
- /sbin：s就是Super User的意思，这里存放的是系统管理员使用的**系统管理程序。**
- /selinux： 这个目录是Redhat/CentOS所特有的目录，Selinux是一个安全机制，类似于windows的**防火墙**，但是这套机制比较复杂，这个目录就是存放selinux相关的文件的。
- /srv：该目录存放一些**服务启动之后需要提取的数据。**
- /sys： 这是linux2.6内核的一个很大的变化。该目录下安装了**2.6内核中新出现的一个文件系统 sysfs 。**sysfs文件系统集成了下面3种文件系统的信息：针对进程信息的proc文件系统、针对设备的devfs文件系统以及针对伪终端的devpts文件系统。该文件系统是内核设备树的一个直观反映。当一个内核对象被创建的时候，对应的文件和目录也在内核对象子系统中被创建。
- /tmp：这个目录是用来存放一些**临时文件**的。
- /usr： 这是一个非常重要的目录，**用户**的很多应用程序和文件都放在这个目录下，类似于windows下的program files目录。
- /usr/bin：系统用户使用的**应用程序。**
- /usr/sbin：超级用户使用的比较**高级的管理程序和系统守护程序。**
- /usr/src：**内核**源代码默认的放置目录。
- /var：这个目录中存放着在不断扩充着的东西，我们习惯将那些**经常被修改的目录**放在这个目录下。包括各种日志文件。
- /run：是一个**临时文件系统**，存储系统启动以来的信息。当系统重启时，这个目录下的文件应该被删掉或清除。如果你的系统上有 /var/run 目录，应该让它指向 run。



基本命令

1、命令名 --help ： 用于获取某个命令的用法帮助。

2、man 1 命令名：获取某命令的使用手册帮助。（和help作用相同）

3、ls：查看文件信息

a、ls -a：显示指定目录下的所有子目录和文件（包括隐藏文件）；

b、ls -l：以列表的方式显示指定目录下的所有子目录和文件的详细信息；

c、ls -h：配合-l以人性化的方式显示文件大小

4、tab键补全命令：有些命令忘记全名了可以只输入前几个字符然后按tab键自动补全。比如ls只输个l，然后按tab键。

5、history：查看历史记录

6、> :重定向命令，将命令的结果重定向到指定文件

7、more：分屏显示，查看内容时，在信息过长无法在一屏上显示时，会出现快速滚屏，使得用户无法看清文件的内容，此时可以使用more命令，每次只显示一页，按下空格键可以显示下一页，按下q键退出显示，按下h键可以获取帮助。

8、|：管道命令，前一个命令的输出作为后一个命令的输入，管道我们可以理解现实生活中的管子，管子的一头塞东西进去，另一头取出来，这里“|”的左右分为两端，左端塞东西(写)，右端取东西(读)。

显示出所有的进程，并通过管道搜索出包含ssh字串的进程

9、clear：清屏命令。（没有实现真正意义的清屏，相当于往下翻了一个空白页）

10、cd：切换目录（绝对路径和相对路径）

相对路径

../切换到上层目录；./切换到当前目录；cd ~切换到主目录；cd -切换到上次进入的路径；/切换到根目录。

绝对路径

11、pwd：显示当前路径

12、mkdir：创建一个新的目录。参数-p可递归创建目录。需要注意的是新建目录的名称不能与当前目录中已有的目录或文件同名，并且目录创建者必须对当前目录具有写权限。

单纯的mkdir命令只能创建一个新的目录，不能递归地创建目录

最后加上参数-p可以递归地创建目录

13、rmdir：删除一个目录。必须离开目录，并且目录必须为空目录，不然提示删除失败。

14、rm：删除文件或目录。使用rm命令要小心，因为文件删除后不能恢复。为了防止文件误删，可以在rm后使用-i参数以逐个确认要删除的文件。

-r：递归删除目录

-i：以交互式方式执行，提示用户是否进行指定的删除操作

-f：强制删除，忽略不存在的文件，无需提示

当前目录下并不存在a,使用参数-f强制删除a也不会报错

15、ln：建立链接文件，链接文件类似于Windows下的快捷方式。链接文件分为软链接和硬链接。

软链接：软链接不占用磁盘空间，源文件删除则软链接失效。ln -s 源文件 链接文件

硬链接：硬链接只能链接普通文件，不能链接目录。两个文件占用相同大小的硬盘空间，即使删除了源文件，链接文件还是存在， ln 源文件 链接文件

注意：如果软链接文件和源文件不在同一个目录，源文件要使用绝对路径，不能使用相对路径。

16、cat：查看文件内容

17、grep:文本搜索，grep允许对文本文件进行模式查找。如果找到匹配模式，grep打印包含模式的所有行。grep一般格式为：grep [-选项] ‘搜索内容串’ 文件名

在grep命令中输入字符串参数时，最好引号或双引号括起来。例如：grep ‘a ’ 1.txt

-v：显示不包含匹配文本的所有行（相当于取反）

-n：显示匹配行及行号

-i：不区分大小写

18、find：查找文件，通常用来在特定的目录下搜索符合条件的文件，也可以用来搜索特定用户属主的文件。

19、cp：拷贝文件，将给出的文件或目录复制到另一个文件或目录中，相当于DOS下的copy命令

20、mv：移动文件，使用mv命令来移动文件或目录，也可以给文件或目录重命名。（此处的重命名值得是复制一份内容相同名字不同的文件出来）

21、tar：归档管理，计算机中的数据经常需要备份，tar是Unix/Linux中最常用的备份工具，此命令可以把一系列文件归档到一个大文件中，也可以把档案文件解开以恢复数据。

tar使用格式 tar [参数] 打包文件名 文件

tar命令很特殊，其参数前面可以使用“-”，也可以不使用。

22、ps：查看进程信息

23、top：动态显示进程

会不断变化

24、kill：终止进程

25、ifconfig：查看网卡信息

26、ping：测试远程主机连通性

27、whoami：查看当前用户

who：查看所有登录系统的用户

28、exit：如果是图形界面，退出当前终端；如果是使用ssh远程登录，退出登陆账户；

如果是切换后的登陆用户，退出则返回上一个登陆账号。

29、useradd：添加用户账号

Linux每个用户都要有一个主目录，主目录就是第一次登陆系统，用户的默认当前目录(/home/用户)；每一个用户必须有一个主目录，所以用useradd创建用户的时候，一定给用户指定一个主目录；用户的主目录一般要放到根目录的home目录下，用户的主目录和用户名是相同的；如果创建用户的时候，不指定组名，那么系统会自动创建一个和用户名一样的组名。

30、which:查看命令所在

31、whoami：查看当前系统当前账号的用户名。可通过cat /etc/passwd查看系统用户信息。

32、who：用于查看当前所有登录系统的用户信息

33、exit：退出当前用户

如果是图形界面，退出当前终端；

如果是使用ssh远程登录，退出登陆账户；

如果是切换后的登陆用户，退出则返回上一个登陆账号

34、useradd：添加新用户。adduser或useradd命令，因为adduser命令是指向useradd命令的一个链接，因此，这两个命令的使用格式完全一样。

35、passwd:设置用户密码

36、userdel：删除用户

37、su:切换用户（可以通过su命令切换用户，su后面可以加“-”。su和su –命令不同之处在于，su -切换到对应的用户时会将当前的工作目录自动转换到切换后的用户主目录）

38、groupadd、groupdel：添加、删除组账号 （cat /etc/group查看用户组）

39、usermod:修改用户所在组（使用方法：usermod -g 用户组 用户名）

-g用来制定这个用户默认的用户组；-G一般配合'-a'来完成向其它组添加

40、groups：查看用户在哪些组（groups 用户名）

41、为普通用户添加sudo权限：新创建的用户，默认不能sudo，需要进行一下操作

sudo usermod -a -G adm用户名；　　sudo usermod -a -G sudo用户名

42、查看有哪些用户组。（方法一:cat /etc/group方法二:groupmod +三次tab键）

43、chmod:修改文件权限。

44、chown:修改文件所有者。

45、修改文件所属组。

46、关机重启：

常用的APT命令参数：

apt-cache search package搜索软件包

apt-cache show package获取包的相关信息，如说明、大小、版本等

sudo apt-get install package安装包

sudo apt-get install package --reinstall重新安装包

sudo apt-get -f install修复安装

sudo apt-get remove package删除包

sudo apt-get remove package --purge删除包，包括配置文件等

sudo apt-get update更新源

sudo apt-get upgrade更新已安装的包

sudo apt-get dist-upgrade升级系统

apt-cache depends package了解使用该包依赖那些包

apt-cache rdepends package查看该包被哪些包依赖

sudo apt-get build-dep package安装相关的编译环境

apt-get source package下载该包的源代码

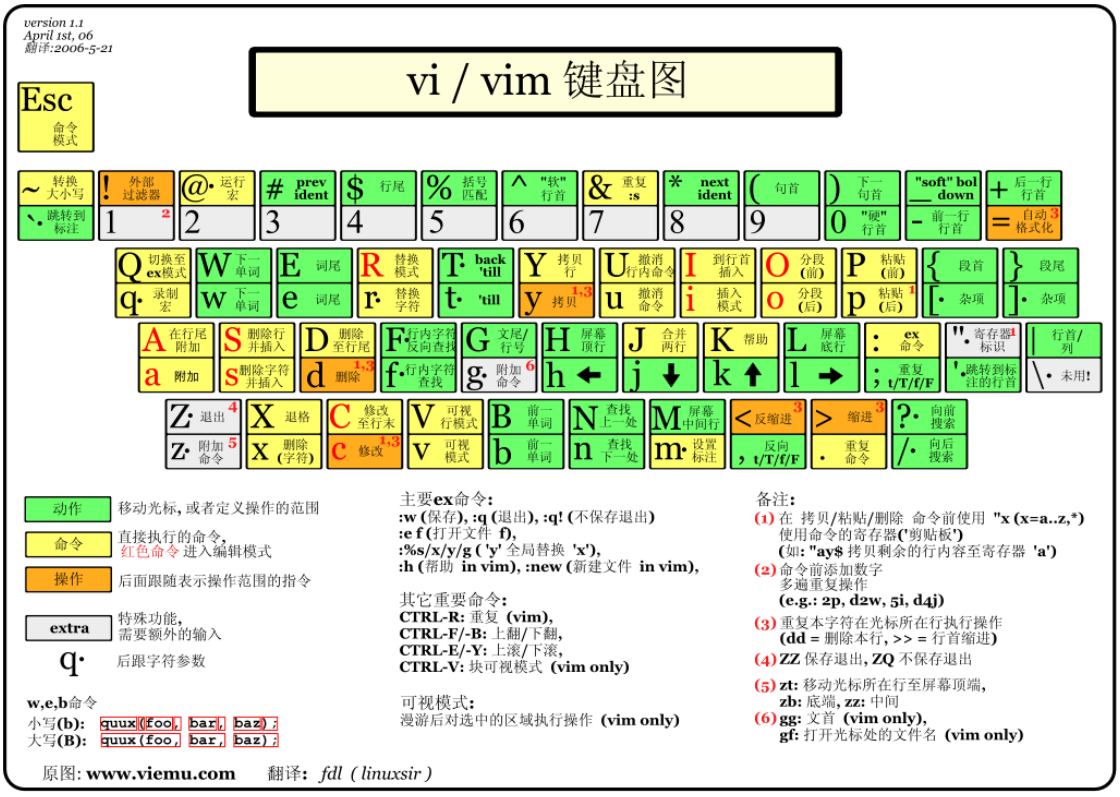
sudo apt-get clean && sudo apt-get autoclean清理无用的包

sudo apt-get check检查是否有损坏的依赖

花一个小时学习一下 Vim,因为你迟早会用它。

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。简单的来说， vi 是老式的字处理器，不过功能已经很齐

全了，但是还是有可以进步的地方。 vim 则可以说是程序开发者的一项很好用的工具。连 vim 的官方网站 (http://www.vim.org) 自己也说 vim 是一个程序开发工具而不是文字处理软件。



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）和底线命令模式（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：

- 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。
- ：（冒号）：切换到底线命令模式，以在最底一行输入命令。

若想要编辑文本：启动Vim，进入了命令模式，按下i，切换到输入模式。
命令模式只有一些最基本的命令，因此仍要依靠底线命令模式输入更多命令。

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：

- 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

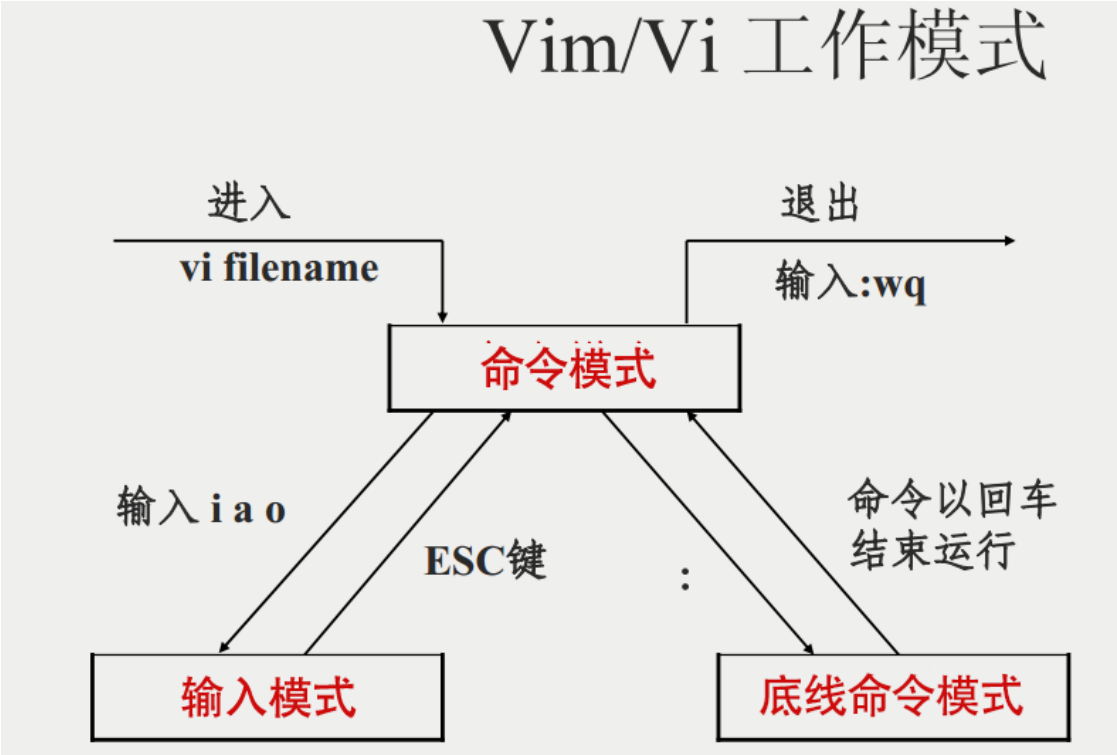
- HOME/END，移动光标到行首/行尾
- Page Up/Page Down，上/下翻页
- Insert，切换光标为输入/替换模式，光标将变成竖线/下划线
- ESC，退出输入模式，切换到命令模式底线命令模式在命令模式下按下:（英文冒号）就进入了底线命令模式。

底线命令模式

可以输入单个或多个字符的命令，可用的命令非常多。在底线命令模式中，基本的命令有（已经省略了冒号）：

- q 退出程序
- w 保存文件

按ESC键可随时退出底线命令模式。简单的说，我们可以将这三个模式想成底下的图标来表示：



第二单元

g++命令有那些参数？怎么填写参数可以更改生成的程序文件名?

> gcc是Linux/Unix上非常强大的C语言编译器。并且如今也支持C++的编译。与VC不同，gcc是完全跨平台的。到目前为止很多工具都使用gcc作为其编译环境，比如cocos2d-x，android的NDK等等。 学习使用gcc编译器，就是学习它众多的编译参数。就是用连字符-连接的参数。

> 因为它的参数并不全是单字符的，所以不能使用一个连字符后加多个参数的写法，要每个参数前面都要加一个连字符

	gcc	g++	vc
main	√	x	x
void main	x	x	√
int main	√	√	√

直接开始

直接在gcc后面加上要编译的C语言源文件

```
gcc hello.c
```

这种默认的情况下会生成一个名为`a.out`的可执行文件。

> 注意，在Linux系统下，后缀名真的不是很重要的事，甚至没有后缀名都可以。这些都没有关系，比如你写一个shell的脚本，其实加不加后缀都是可以运行的。此时后缀的目的是为了便于用户管理和区分文件而已。By the way，如果你执行`./hello.c`，那么系统会把它也当做shell脚本来运行，根本不管什么后缀。

-o

最常用的编译选项，用于指定要生成的可执行文件的名称。

```
gcc -o hello hello.c
```

需要体会的一点是：gcc对于参数的位置无要求！比如也可以写作

```
gcc hello.c -o hello
```

惟一要注意的就是`-o`后面一定要紧跟要生成的可执行文件名。

> 编译出的可执行文件，在Windows下就是exe（executive）。但Linux下，后缀名无限制。一般不指定后缀就可以了。

-g

用于给生成的可执行文件加上调试信息，只有这样才可使用<kbd>gdb</kbd>调试。

```
gcc -g -o hello hello.c
```

同样参数-g 的也可写在hello.c后面。

-c

终止链接器的运行，输出文件为汇编后的目标文件*.o

```
gcc -c hello.c
```

生成文件为hello.o

-O2

对源码进行优化，使编译出的程序，运行效率更高。注意是大写的英语字母O ，不是阿拉伯数字0

```
gcc -O2 hello.c -o hello
```

O 是<kbd>Optimize</kbd>之意。同样还有O1 ，但是优化效果不如O2 ，缺省是O0

-D

给编译的源文件传递一个宏。

```
gcc a.c -DHELLO -DWORLD=10
```

相当于：

```
//在a.c中定义了
# define HELLO
# define WORLD 10
```

-E

gcc -E 指示gcc对源代码进行预处理，结果直接输出到终端。

> 实际上和命令cpp 源文件 相同。cpp是c预处理器的意思，而非c++的意思。

-I

包含自定义头文件的路径

-S

生成汇编代码以.s为后缀。
默认是**AT&T**汇编语法，加选项**masm=intel**可生成**Intel**语法的汇编。

```
gcc -S -masm=intel test.c
```

链接

静态链接库

静态链接库是后缀名为.a的文件。它有多个后缀为你.o的目标文件组成。

使用-c 参数可以使编译在链接前终止，所以生成的是源文件对应的目标文件。

```
gcc -c addvec.c multvec.c
ar rcs libvector.a addvec.o multvec.o
```

ar 是archive档案的缩写。上面命令生成了.a的静态库文件，在链接时，要如下：

```
gcc -c main2.c
gcc -static -o p2 main2.o ./libvector.a
```

动态链接库

后缀名为.so的是动态共享链接库文件，其中的s就是shared共享的意思，如下命令：

```
gcc -shared -fPIC -o libvector.so addvec.c multvec.c
gcc -o p2 main2.c ./libvector.so
```

完成了生成.so以及链接.so的操作。<kbd>-fPIC</kbd>指示生成与位置无关的代码。

-I

```
gcc -o temp temp.c -lm
```

-lm选项，在编译时会进入系统库路径搜索，链接“数学库”。常用的库会自动链接，无需指定。
系统缺省的库路径为：/lib、/usr/lib、/usr/local/lib、/usr/lib64。

> 数学库的文件可能为*libm-2.1.2.so*去掉lib和后面的版本号就只剩下m了。

-L

如果该库不在系统缺省路径下（比如第三方库，自定义的库），还要使用**-L**选项指定路径。

```
-L/home/jelly/mylib
```

-Wl,-rpath

-L选项指定的是在编译期间库的搜索路径，然而如果是动态库的话，在运行时加载库，此时只会搜索默认库路径。
此时需在编译时加上**-Wl,-rpath=**后面指定路径。注意这不是两个选项。

```
g++ -I ../include -Wl,-rpath=../util/ -lwang -L../util/  client.c -o client
```

另外一种方案是修改配置文件：`/etc/ld.so.conf`。然后需要**ldconfig**命令（root）更新。
如果无root权限，请使用上一种方案。

```
有时候连接通过，但运行时出错：未定义符号XXX。请查看一下是否正确保护库的路径，如果确认无误，仍有错。

可以使用**ldd**命令检查链接的库的绝对路径。

有时候自己写的库和系统的库重名就会出现该错误。
```

使用c99标准编译

默认的情况下，编译器是以c89的标准编译的。使用c99则：

```
gcc -std=c99 hello.c
```

包含非系统的头文件

-I 选项指定头文件的位置。一般和L选项联用。比如：

```
gcc mysql_test.c -I /usr/local/include/mysql -L/usr/local/lib -lmysqlclient -
o test
```

-M检查所需的头文件

```
gcc -M main.c
```

以makefile风格显示源文件的依赖关系。会列出所有包含的所有头文件。会列出标准库的头文件。
如果不想显示标准库的头文件，请使用**-MM**选项代替**-M**。

其他

-fno-elide-constructors

适用于g++。C++语言因为各种临时对象的问题，所以编译器通常会自行进行优化，比如**NRV**优化（O0已存在该优化），会减少几次拷贝构造函数的调用过程。如果你想关闭这个优化：

```
g++ -fno-elide-constructors hello.cpp
```

完善章节中的Hello SLAM小程序，并把它做成一个小程序库，安装到本地硬盘中。然后新建一个工程，找到这个库并调用。

```
//hello slam
#include<iostream>
using namespace std;
int main(int argc, char **argv) {
    cout << "Hello SLAM!" << endl;
    return 0;
}
```

封装成库:

```
//libslam.cpp
#include<iostream>
using namespace std;
void printHello() {
    cout << "Hello SLAM!" << endl;
}
```

```
//libslam.h
#ifndef LIBHELLOSLAM_H_
#define LIBHELLOSLAM_H_

void printHello();

#endif
```

调用程序:

```
//useHello
#include "libHelloSLAM.h"
int main(int argc, char **argv) {
    printHello();
    return 0;
}
```

CMakeList.txt:

```
# 声明版本
cmake_minimum_required(VERSION 2.8)

# 声明 cmake 工程
project(HelloSLAM)

# 添加一个可执行程序
# 语法： add_executable( 程序名 源代码文件 )
add_executable(helloSLAM helloSLAM.cpp)

# 添加一个静态库
add_library(hello libHelloSLAM.cpp)
```

```
# 添加一个共享库
add_library(hello_shared SHARED libHelloSLAM.cpp)

# 使用共享库
add_executable(useHello useHello.cpp)
target_link_libraries(useHello hello_shared)
```

阅读《cmake实践》，了解cmake的其他语法

下载链接：cmake实践

寻找其他cmake教学材料，深入了解cmake，例如cmake-tutorial

CMake 入门实战

Cmake如何入门-知乎

第三单元