



## 第2章 linux基础





# 本章内容

- 了解Linux操作系统的结构
- Linux图形桌面和命令行界面
- 掌握登录和退出过程
- 简要介绍一些常用的shell
- 介绍一些初学者会用到的命令
- 简要介绍shell中的一些元字符
- 掌握几个常用文本编辑器
- 命令echo, exit, hostname, ls, man, passwd, set, setenv, uname, whatis, whereis, who, whoami, alias, cal, cat, cd, mkdir, pwd, rmdir, uptime





## 2.1 Linux系统架构





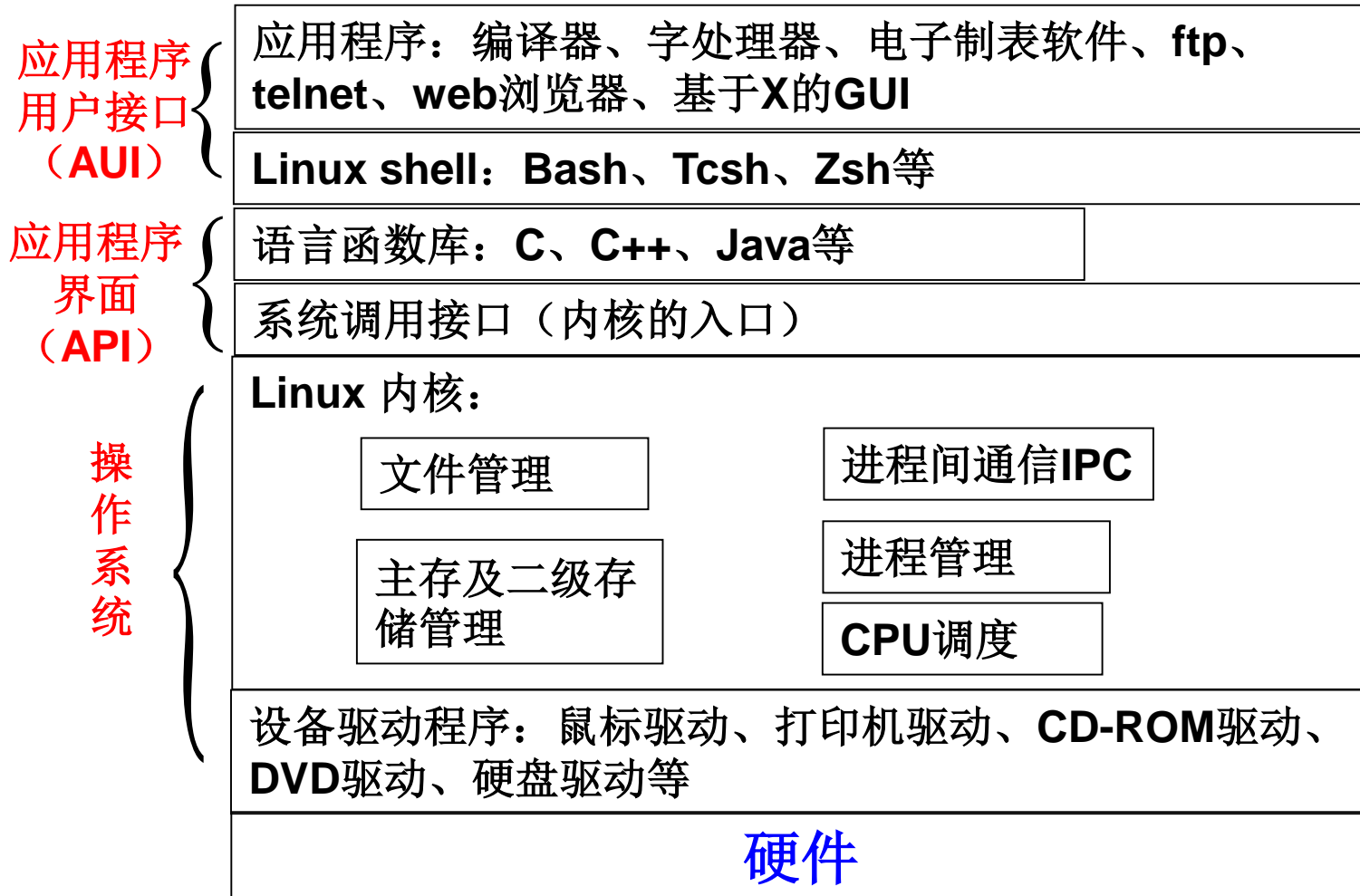
# Linux系统架构

|   |                              |  |   |   |                         |  |
|---|------------------------------|--|---|---|-------------------------|--|
| User mode   | User applications            | For example, <code>bash</code> , LibreOffice, Apache OpenOffice, Blender, 0 A.D., Mozilla Firefox, etc.  |   |   |                         |  |
|   | Low-level system components: | System daemons:<br><code>systemd</code> , <code>runit</code> , <code>logind</code> ,<br><code>networkd</code> , <code>soundd</code> , ...  | Windowing system:<br><code>X11</code> , <code>Wayland</code> , <code>Mir</code> ,<br><code>SurfaceFlinger</code><br>(Android) | Other libraries:<br><code>GTK+</code> , <code>Qt</code> , <code>EFL</code> , <code>SDL</code> , <code>SFML</code> , <code>FLTK</code> , <code>GNUstep</code> , etc. |                         | Graphics:<br><code>Mesa</code> ,<br><code>AMD Catalyst</code> ,<br>... |
|   | C standard library           | <code>open()</code> , <code>exec()</code> , <code>sbrk()</code> , <code>socket()</code> , <code>fopen()</code> , <code>calloc()</code> , ... (up to 2000 subroutines)<br><code>glibc</code> aims to be POSIX/SUS-compatible, <code>uClibc</code> targets embedded systems, <code>bionic</code> written for Android, etc.   |   |   |                         |  |
| Kernel mode   | Linux kernel                 | <code>stat</code> , <code>splice</code> , <code>dup</code> , <code>read</code> , <code>open</code> , <code>ioctl</code> , <code>write</code> , <code>mmap</code> , <code>close</code> , <code>exit</code> , etc. (about 380 system calls)<br>The Linux kernel System Call Interface (SCI, aims to be POSIX/SUS-compatible) |   |   |                         |  |
|   |                              | Process scheduling subsystem   | IPC subsystem   | Memory management subsystem   | Virtual files subsystem | Network subsystem  |
|   |                              | Other components: <code>ALSA</code> , <code>DRI</code> , <code>evdev</code> , <code>LVM</code> , device mapper, Linux Network Scheduler, Netfilter<br>Linux Security Modules: <code>SELinux</code> , <code>TOMOYO</code> , <code>AppArmor</code> , <code>Smack</code>  |   |   |                         |  |
| Hardware (CPU, main memory, data storage devices, etc.) |                              |  |   |   |                         |  |





# Linux软件体系结构





## 2.2 Linux图形桌面和命令行界面

---





## 2.2 Linux图形桌面和命令行界面

- LINUX系统是**多进程**、**多用户**和**交互式**的计算环境。
- 使用Linux系统的方式
  - 基于图形用户界面( graphic user interface, GUI)
  - 基于文本的界面(command line interface, CLI, 命令行界面)  
telnet、ssh等软件（命令）用于远程登录





# 基于图形用户界面（GUI）

## ■ Linux当前比较流行的图形桌面

- KDE桌面
- GNOME桌面
- Unity桌面





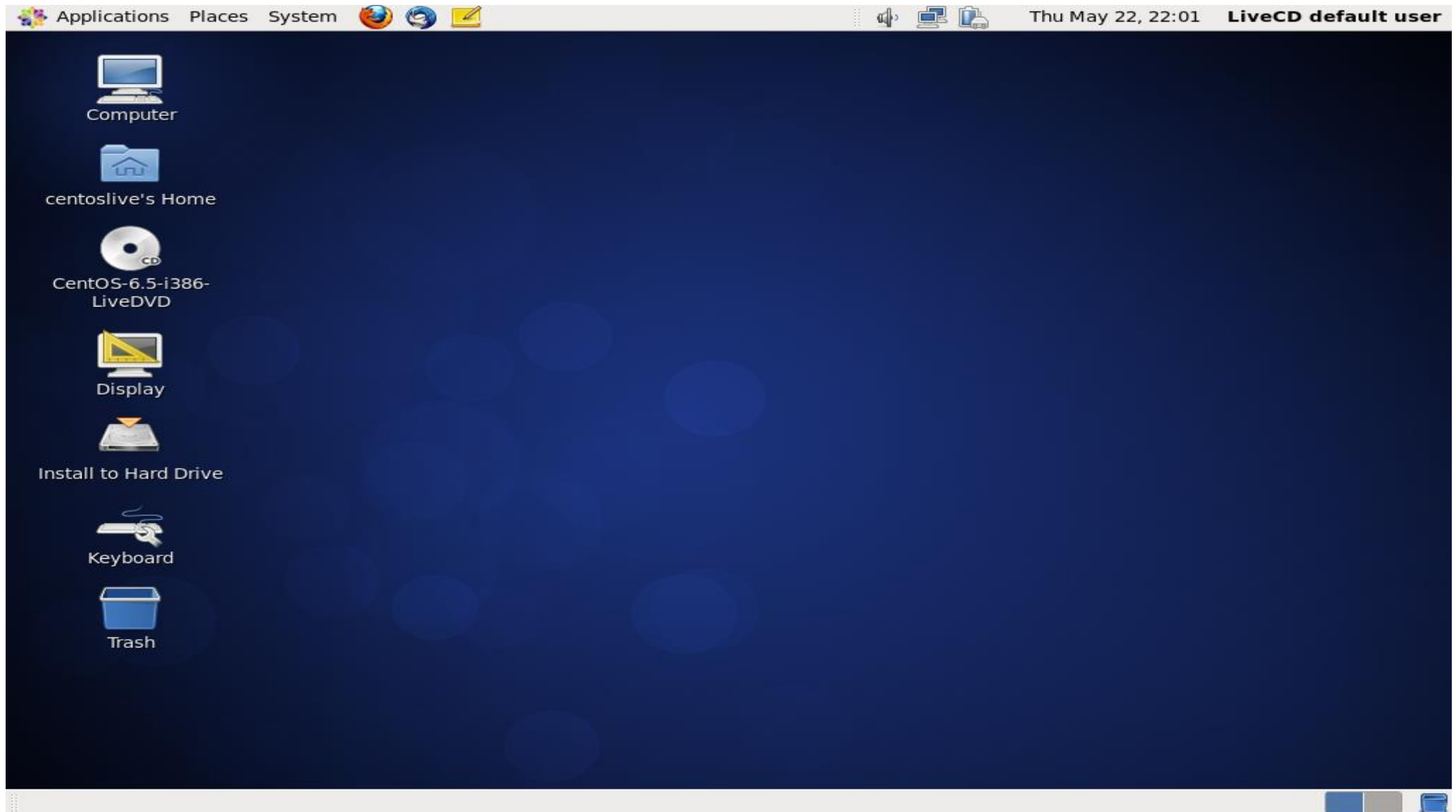


# KDE



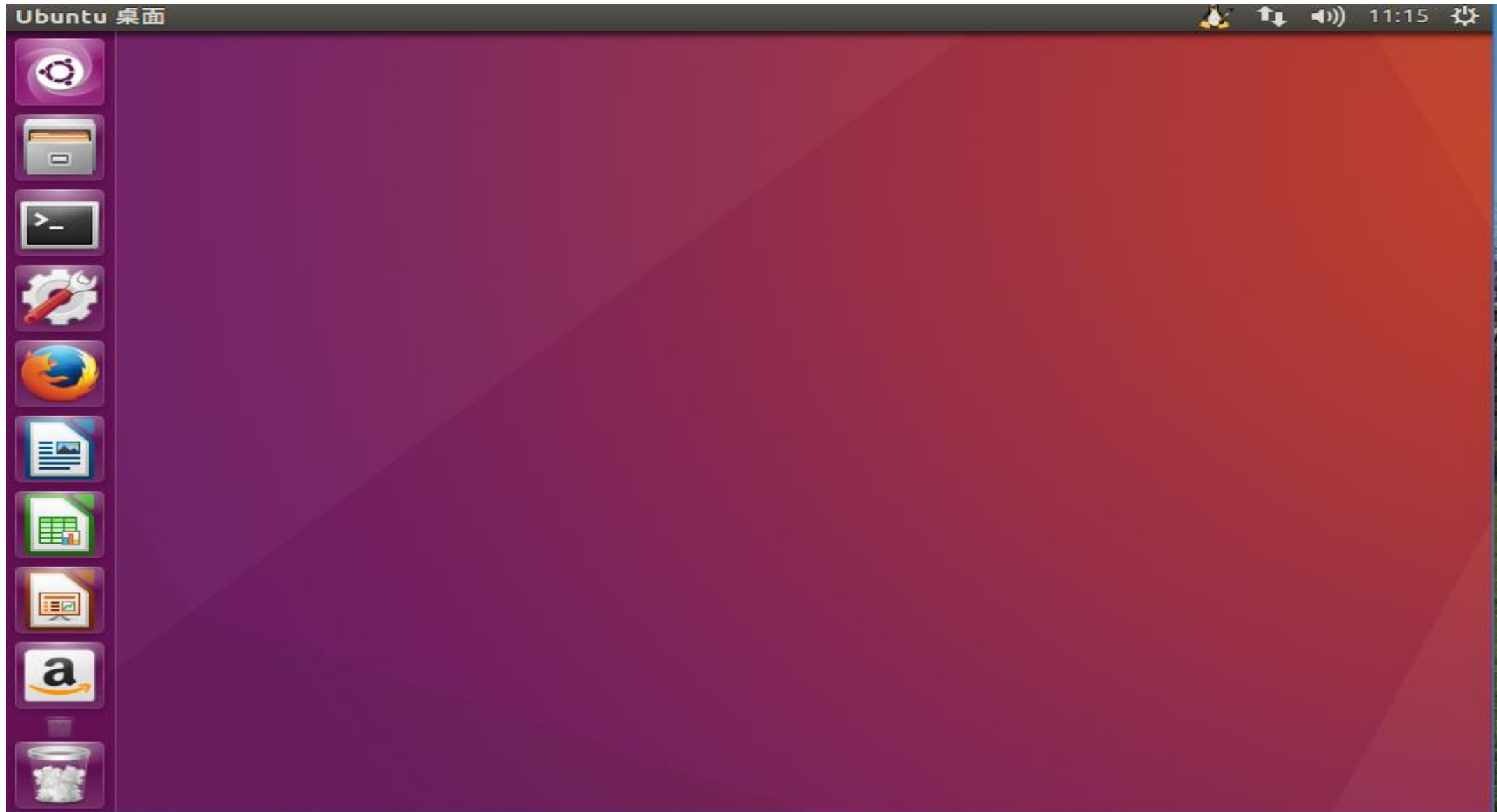


# GNOME



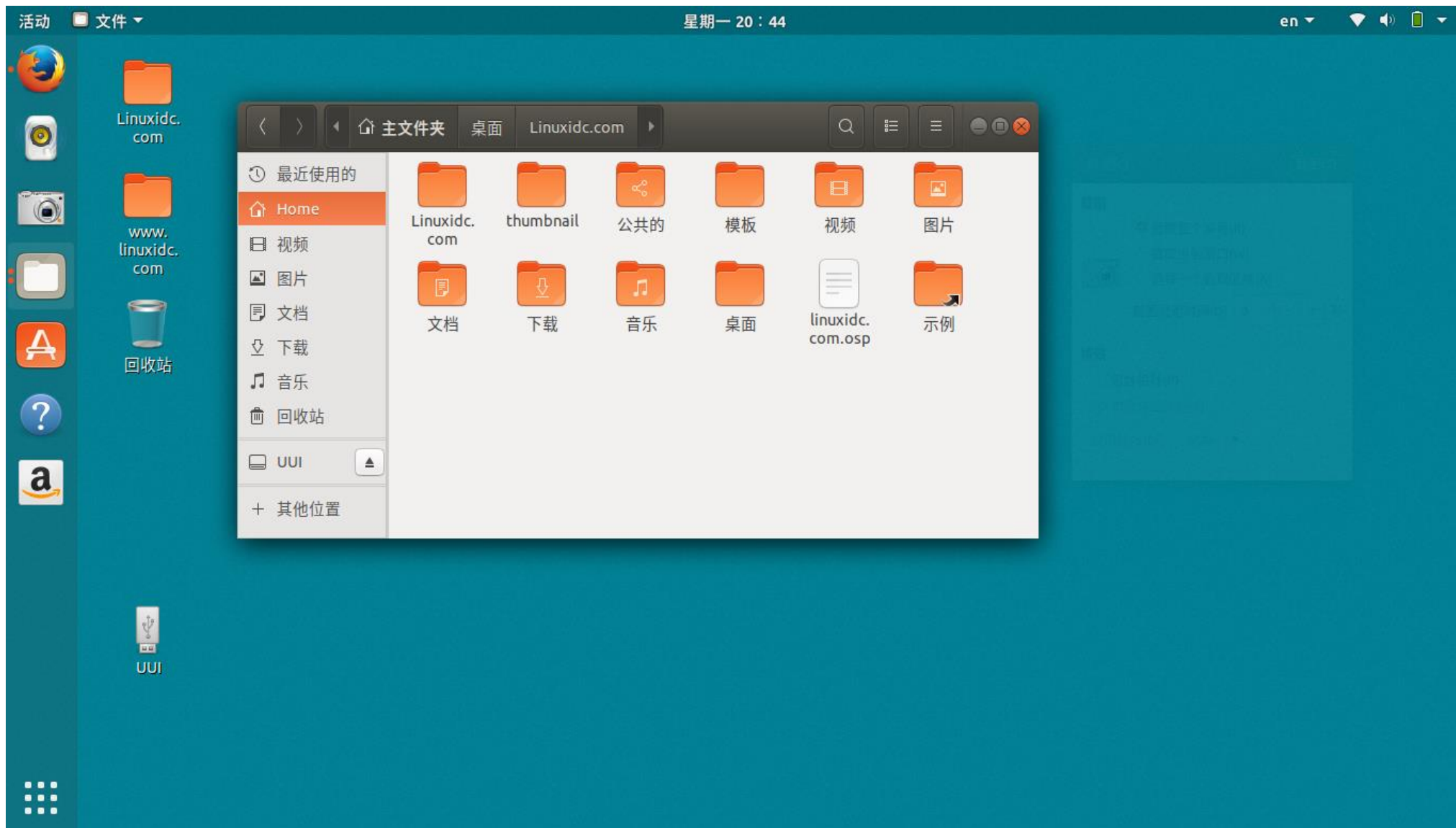


# Unity (Ubuntu)





# Ubuntu 17.10开始使用GNOME桌面





# 如何使用Linux命令行

- 在图形化桌面出现之前，与Unix系统进行交互的唯一方式就是借助由shell所提供的**文本命令行界面（command line interface, CLI）**。CLI只能接受文本输入，也只能显示出文本和基本的图形输出。
- **Linux控制台**：进入CLI的一种方法是让Linux系统退出图形化桌面模式，进入文本模式。这样在显示器上就只有一个简单的shell CLI，跟图形化桌面出现以前一样。
- **Linux虚拟控制台**：虚拟控制台是运行在Linux系统内存中的终端会话。系统启动GUI后，按组合键<**Ctrl+Alt+(F1~F7)**>





# 如何使用Linux命令行

- **Linux仿真终端**：在Linux桌面图形化窗口中模拟控制台终端的使用。

建议使用这种方式

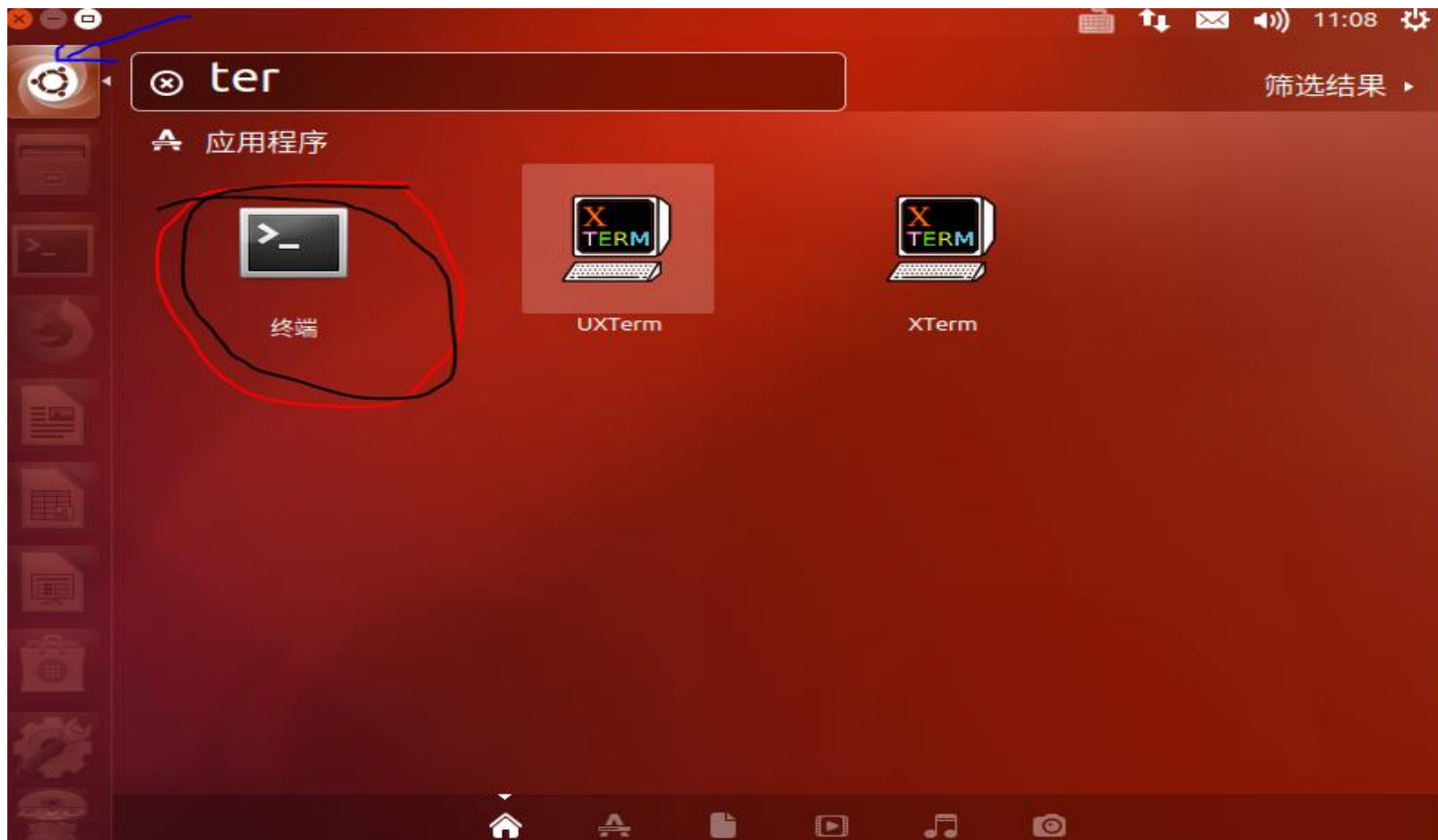
- **GNOME Terminal 仿真器**：GNOME Terminal软件包是GNOME桌面环境的默认终端仿真器。很多发行版，如RHEL、Fedora和CentOS，默认采用的都是GNOME桌面环境。**Ubuntu Unity**，也采用**GNOME Terminal**作为默认的终端仿真软件包。





# Ubuntu打开Linux命令行仿真终端

- 第一种方法：使用快捷键Ctrl+Alt+T快速访问GNOME终端。
- 第二种方法：Applications ⇨ System Tools ⇨ Terminal 或 “应用程序” ⇨ “终端”





# 远程连接到Linux系统的方式

- 一般使用远程登录软件连接互联网的一台Linux主机或服务器上。在Windows下使用的远程登录软件有PuTTY、telnet等，如：在<http://www.putty.org/>下载软件。
- 配置Linux服务器的SSH服务，以Ubuntu为例：  
`sudo apt install openssh-server`
- 查看sshd服务：`ps -e | grep sshd`
- 启动sshd服务：`sudo service sshd start`
- 查看Linux服务器IP地址,使用命令：`ifconfig`







# 远程连接到Linux系统的方式

- PuTTY的设置，如：
  - host: 192.168.217.131 , user:user1, port:22, password:
- 连接Linux系统时，需要向系统输入正确的用户名和口令。在成功登录到Linux计算机后，屏幕会出现一个诸如“\$”(有可能是其他的符号)的shell提示符。接着用户可以使用各种各样的linux命令了。

**Login : your username**

**Password: your password**





## PuTTY Configuration



Category:

- [-] Session
  - .... Logging
- [-] Terminal
  - .... Keyboard
  - .... Bell
  - .... Features
- [-] Window
  - .... Appearance
  - .... Behaviour
  - .... Translation
  - .... Selection
  - .... Colours
- [-] Connection
  - .... Data
  - .... Proxy
  - .... Telnet
  - .... Rlogin
  - [+] SSH
    - .... Serial

### Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address)

192.168.217.131

Port

22

Connection type:

☐ Raw

☐ Telnet

☐ Rlogin

☒ SSH

☐ Serial

Load, save or delete a stored session

Saved Sessions

|

Default Settings

ub17

Load

Save

Delete

Close window on exit:

☐ Always

☐ Never

☒ Only on clean exit

About

Open

Cancel





user1@ubuntu: ~

login as: user1

user1@192.168.217.131's password:

Welcome to Ubuntu 18.04 LTS (GNU/Linux 4.15.0-46-generic x86\_64)

- \* Documentation: <https://help.ubuntu.com>
- \* Management: <https://landscape.canonical.com>
- \* Support: <https://ubuntu.com/advantage>

- \* Canonical Livepatch is available for installation.
  - Reduce system reboots and improve kernel security. Activate at:  
<https://ubuntu.com/livepatch>

395 个可升级软件包。

9 个安全更新。

Last login: Wed Mar 13 12:18:08 2019 from 192.168.217.1

To run a command as administrator (user "root"), use "sudo <command>".

See "man sudo\_root" for details.

user1@ubuntu:~\$





## 2.3 启动、关机及shell

---






# Linux启动模式

- Linux系统的运行级别（启动级）从 0-6 共7个。
  - 0 为**停机**，关闭系统。
  - 1 为**单用户模式**，就像Windows下的安全模式类似。
  - 2 为**多用户模式**，但是没有NFS 支持。
  - 3 为**完整的多用户模式，是标准的运行级**。
  - 4 **保留**，在一些特殊情况下可以用它来做一些事情。例如在笔记本电脑的电池用尽时，可以切换到这个模式来做一些设置。
  - 5 X Window 系统了。
  - 6 为**重新重启**，运行 init 6 机器就会重启。startx命令
- 运行级配置文件放在/etc/inittab中（ubuntu没有此文件），有一行“id:**5**:initdefault”
- root身份在终端上执行**telinit n**，进入运行级n。





# Linux退出命令

- 文本界面（命令行）启动，用户退出系统  
按<Ctrl-D>键或logout命令
- 图形界面，用户退出系统，Ubuntu：
  - 鼠标点击 “ →关  ”
- 关机命令：
  - 命令行方式：shutdown, halt, init 0, poweroff等，需要root权限
  - 图形桌面：鼠标点击 “桌面→关机”

“学在浙大” → 理解 Linux 中的 shutdown、poweroff、halt 和 reboot 命令





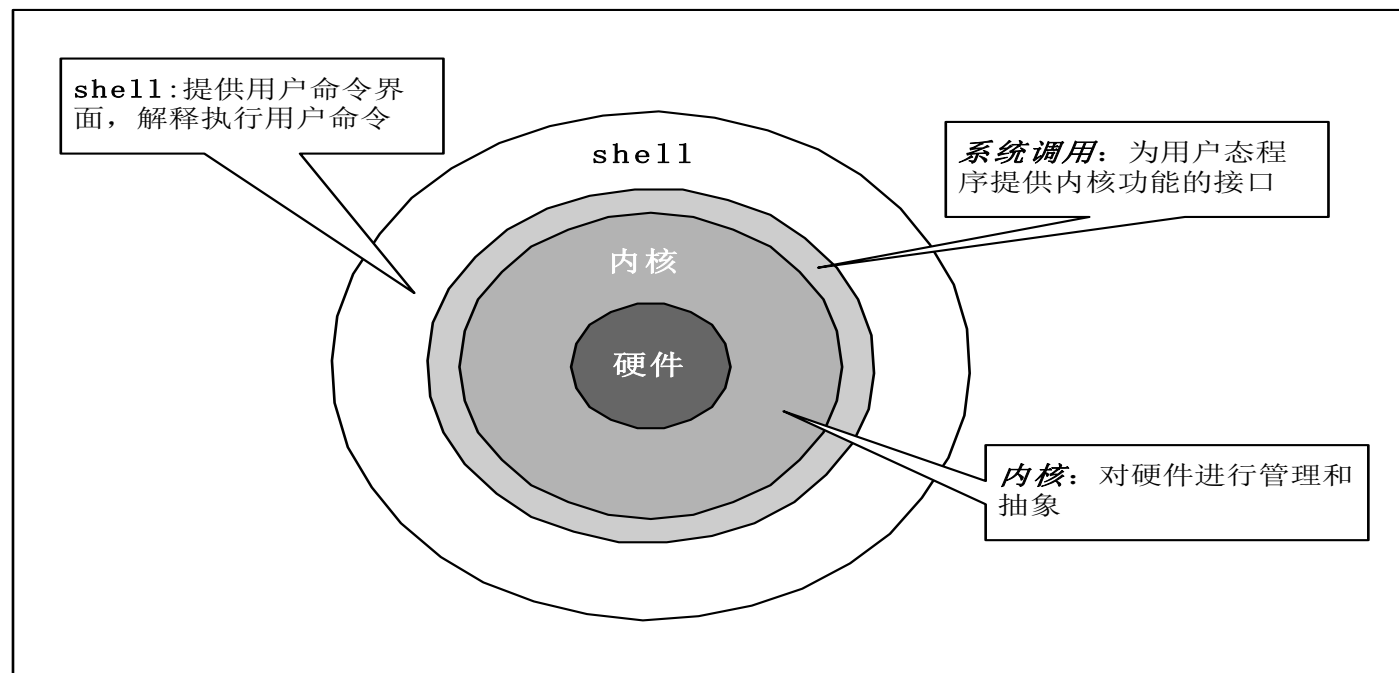
# 关机命令

- 使用 shutdown 或 halt 命令关闭Linux系统。当然使用这些命令需要有管理员用户权限。
- 例：下面是指定在早上 8:00 关机。  
\$ shutdown -h 8:00
- 例：下面的命令是指定计算机在三分钟后关机。  
\$ shutdown -h +3
- 例：下面的命令是指定计算机立刻关机。  
\$ halt
- 例：下面的命令使计算机重新开机。  
\$ reboot  
\$ init 6



# shell简介

- **shell** 是Linux系统的用户界面，提供了用户与内核进行交互操作的一种接口。它为用户提供了启动程序、管理文件系统中的文件以及运行在Linux系统上的进程的途径。
- shell也被称为Linux的命令解释器(command interpreter)







# shell简介

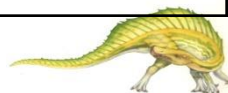
- 在Linux系统上，通常有多种Linux shell可用。不同的shell有不同的特性，有些更利于创建脚本，有些则更利于管理进程。
- 所有Linux发行版默认的shell都是**bash shell**。bash 由GNU项目开发，被当作标准Unix shell——Bourne shell（以创建者的名字命名）的替代品。bash shell的名称就是针对Bourne shell的拼写所玩的一个文字游戏，称为**Bourne again shell**。





# 常用的shell

| shell名称       | 相关历史  |
|---------------|---|
| sh ( Bourne ) | 源于UNIX早期版本的最初的shell   |
| cshtcsh, zsh  | C shell及其变体，最初是由Bill Joy在Berkeley UNIX上编写的。它可能是继bash和Korn shell之后第三个最流行的shell   |
| ksh, pdksh    | korn shell和它的公共域兄弟pdksh（public domain korn shell）由David Korn编写，它是许多商业版本UNIX的默认shell   |
| bash          | 来自GNU项目的bash或Bourne Again Shell是Linux的主要shell。它的优点是 <code>可以免费获取其源代码，即使你的UNIX系统目前没有运行它，它也很可能已经被移植到该系统中。</code> bash与Korn shell有许多相似之处 |





# 常用的shell

| shell名称            | 存放的位置          | 程序名  |
|--------------------|----------------|------|
| Bourne shell       | /bin/sh->bash  | bash |
| Bourne Again shell | /bin/bash      | bash |
| C shell            | /bin/csh->tcsh | tcsh |
| TC shell           | /bin/tcsh      | tcsh |
| Korn shell         | /bin/ksh       | ksh  |

■ Shell命令可以被分为 **内部(内置builtin)命令**和 **外部命令**。

- 内部命令是shell本身包含的一些命令，这些内部命令的代码是整个shell代码的一个组成部分；
- 外部命令的代码则存放在一些二进制的**可执行文件**或者**shell脚本**中
- type命令判断是否内部还是外部命令：

\$ type cd

cd 是shell 内建





# shell命令搜索路径

- shell搜索的目录的名字都保存在一个shell变量**PATH**（在TC shell中是path）中。
- 变量**PATH**（或者path）中的目录名用一些特定的符号分开。在bash shell中，目录名用**冒号**分开。
- **\$ echo \$PATH**  
/usr/local/globus/bin:/usr/local/globus/sbin:/usr/java/j2sdk1.4.1\_01/bin:/usr/local/apache-ant-1.5.4/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/X11R6/bin:/opt/hdf5-oscar-1.4.4-post2/bin:/opt/pbs/bin:/opt/pbs/lib/xpbs/bin:/opt/pvm3/lib:/opt/pvm3/lib/LINUX:/opt/pvm3/bin/LINUX:/opt/env-switcher/bin:/opt/lam-6.5.9/bin:/root/bin
- 变量**PATH**（或者path）保存在**主目录**中的隐藏文件（hidden file）.profile或者.login中





# shell的环境变量

- 对于普通计算机用户，控制台窗口的属性和外观，以及解释命令的环境，都是由系统管理员设定的。环境设置是由**环境变量**控制的。环境变量在用户登录时得到，或事先设成默认值。当用户键入命令或执行其他重要操作时，环境决定了使用哪种shell或命令行解释器。

- 显示shell的路径：

**echo \$SHELL**

注意大小写

- 查看环境变量值：

**set**

- 设置环境变量值：

**变量名=值**

◆ 环境变量 (environment variables)  
一般是指在操作系统中用来指定操作系统运行环境的一些参数





# bash shell支持的Bourne变量

- CDPATH 冒号分隔的目录列表，作为cd命令的搜索路径
- **HOME** 当前用户的主目录
- IFS shell 用来将文本字符串分割成字段的一系列字符
- MAIL 当前用户收件箱的文件名（bash shell会检查这个文件，看看有没有新邮件）
- MAILPATH 冒号分隔的当前用户收件箱的文件名列表（bash shell会检查列表中的每个文件，看看有没有新邮件）
- OPTARG getopt命令处理的最后一个选项参数值
- OPTIND getopt命令处理的最后一个选项参数的索引号
- **PATH** shell查找命令的目录列表，由冒号分隔
- PS1 shell命令行界面的主提示符
- PS2 shell命令行界面的次提示符





## 部分bash shell环境变量

- **BASH** 当前shell实例的全路径名
- **BASH\_COMMAND** shell正在执行的命令或马上就执行的命令
- **BASH\_ENV** 设置了的话，每个bash脚本会在运行前先尝试运行该变量定义的启动文件
- **BASH\_SOURCE** 含有当前正在执行的shell函数所在源文件名的数组变量
- **BASH\_VERSION** 当前运行的bash shell的版本号
- **BASH\_XTRACEFD** 若设置成了有效的文件描述符（0、1、2），则'set -x'调试选项生成的跟踪输出可被重定向。通常用来将跟踪输出到一个文件中
- **OLDPWD** shell之前的工作目录
- **PWD** 当前工作目录
- **RANDOM** 返回一个0~32767的随机数（对其的赋值可作为随机数生成器的种子）
- **SECONDS** 自从shell启动到现在的秒数（对其赋值将会重置计数器）
- **SHELL** bash shell的全路径名
- **UID** 当前用户的真实用户ID（数字形式）





## 部分bash shell环境变量

- BASHPID 当前bash进程的PID
- COMP\_LINE 当前命令行
- COPROC 占用未命名的协进程的I/O文件描述符的数组变量
- ENV 如果设置了该环境变量，在bash shell脚本运行之前会先执行已定义的启动文件（仅用于当bash shell以POSIX模式被调用时）
- EUID 当前用户的有效用户ID（数字形式）
- FUNCNAME 当前执行的shell函数的名称
- GROUPS 含有当前用户属组列表的数组变量
- HISTFILE 保存shell历史记录列表的文件名（默认是.bash\_history）
- HOSTNAME 当前主机的名称
- HOSTTYPE 当前运行bash shell的机器
- LC\_CTYPE 决定如何解释出现在文件名扩展和模式匹配中的字符
- LINENO 当前执行的脚本的行号
- LINES 定义了终端上可见的行数

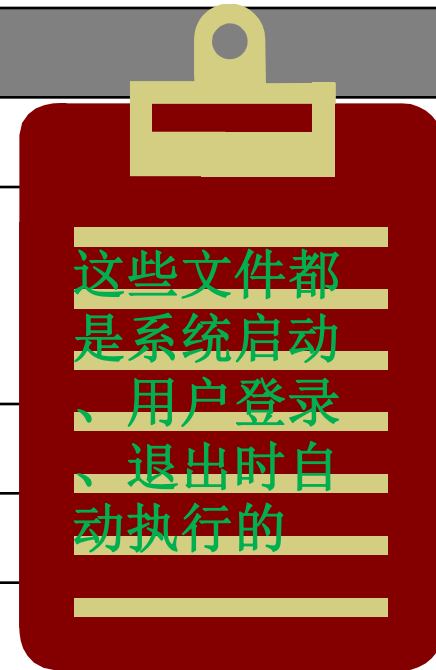






# bash中的一些启动文件

| 表 bash的一些启动文件                              |              |
|--|--------------|
| 文件名  | 功能描述         |
| /etc/profile                               | 登录时自动执行      |
| ~/.bash_profile, ~/.bash_login, ~/.profile | 登录时自动执行      |
| ~/.bashrc                                  | Shell登录时自动执行 |
| ~/.bash_logout                             | 退出时自动执行      |
| ~/.bash_history                            | 记录最近会话中的命令   |
| /etc/passwd                                |              |

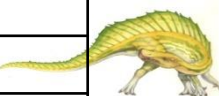




# shell元字符

- shell元字符(shell metacharacters)：除了字母和数字，其它有特殊的含义的大部分字符。

| Shell 中的元字符 |                   |                          |
|-------------|-------------------|--------------------------|
| 元字符         | 作用                | 例子                       |
| 回车换行        | 结束一个命令行           |                          |
| 空格          | 分割命令行中的元素         | ls /etc                  |
| Tab         | 分割命令行中的元素         | ls /etc                  |
| #           | 开始一行注释            | # this is a comment line |
| “           | 引用多个字符但是允许替换（15章） | “\$file”.bak             |
| \$          | 表示一行的结束以及显示变量的值   | \$PATH                   |
| &           | 让一个命令在后台执行        | command &                |
| ‘           | 引用多个字符            | ‘\$100,000’              |
| ()          | 在子shell中执行命令      | (command1;command2)      |
| *           | 匹配0个或者多个字符        | chap*.ps                 |
| []          | 插入通配符             | [a-s] 或者 [1,5-9]         |
| ^           | 表示一行的开始以及作为否定符号   | [^3-8]                   |
| `           | 替换命令              | PS1=`command`            |
| { }         | 在当前shell中执行命令     | {command1;command2}      |
|             | 创建命令间的管道          | command1 command2        |
| ;           | 分割顺序执行的命令         | command1;command2        |





# shell元字符(续)

|   |                                |   |
|---|--------------------------------|---|
| < | 重定向命令的输入                       | <b>command&lt;file</b>                    |
| > | 重定向命令的输出                       | <b>command&gt;file</b>                    |
| ? | 匹配单个字符                         | <b>lab.?</b>                              |
| / | 用作根目录或者路径名中的分割符                | <b>/usr/bin</b>                           |
| \ | 转义字符；转义回车换行字符，允许在下一行中继续shell命令 | <b>command arg1\<br/>arg2 arg3<br/>\?</b> |
| ! | 启动历史记录列表中的命令和当前命令              | <b>!! , !4</b>                            |
| % | TC shell的提示符，或者指定一个任务号时作为起始字符  | <b>% 或者 %3</b>                            |
| ~ | 代表主目录                          | <b>~/.profile</b>                         |





# 通配符

## ■ 常用通配符：\*、?、[]、-

- “[ ]-” 用于构成字符组模式，如：

- ▶ [abc] 表示匹配a、b、c中一个字符
- ▶ [a-z] 表示匹配小写英文字母中的一个字符

## ■ 通配符扩展:花括号{}通配符,允许将任意的字符串分组放在一个集合中，以供shell进行扩展。

- 还可以使用花括号扩展用相关的名字创建子目录：

```
$ ls -F
```

```
file1 fi1e2 fi1e3
```

```
$ mkdir dir{A,B,C,D,E}
```

```
$ ls -F
```

```
file1 fi1e2 fi1e3 dirA/ dirB/ dirC/ dirD/ dirE/
```





# 学会使用<tab>键

- <tab>键:命令行补全

cd d<tab> => cd dir1

- 把当前目录设置为当前目录下d开头的子目录





## 例

- 例：字符串“?.txt”可以用来表示一个字符后跟“.txt”的所有文件名，如：a.txt, 1.txt, @.txt。
- 例：字符串[0-9].c用来表示所有文件名为单个数字后跟“.c”形式的文件，如：1.c和3.c。
- 例：字符串lab1 \ / c表示lab1/c。注意，在这里，我们用反斜线号（\）来处理“消除了特殊意义”的斜线号（/）。
- 例：下面的这条命令显示当前目录中所有由2个字符组成，且以.html为结尾的文件。而且这些文件名的第一个字符是数字，第二个字符是大写或者小写的字母。
  - \$ ls [0-9][a-zA-Z].html





## 2.4 常用的几个命令

---





# 命令语法结构

## ■ 命令行输入的语法结构：

`$ command [[-]option(s)] [option argument(s)] [command argument(s)]`

- \$是来自操作系统的提示符，你的计算机可能不一样。
- 任何括在[ ]中的内容都不是必需的
- command是Linux命令（不同的shell有所区别），小写（命令名）
- [-option(s)]是定制命令动作的一个或多个修饰符号（选项）
- [option argument(s)]是定制选项动作的一个或多个修饰符号（选项的参数）
- [ command argument(s) ] 受命令影响的一个或者多个对象（命令的参数）







# 例

- \$ ls
- \$ ls -la
- \$ ls -la m\*
- \$ lpr -Pspr -n 3 proposal.ps
  - 命令lpr，两个选项：P和n,作为选项的操作对象的两个选项参数：spr和3,以及命令参数：proposal.ps。
  - 这些项都是区分大小写的。
  - 一个选项和它的参数间是有空格的，但是另一个选项和它的参数间是没有空格的





# 常用的几个命令

- **passwd** : 修改密码命令
- **su**     改变用户身份
- **sudo**    以root用户权限运行命令
- 与用户有关的其他命令
  - **useradd/adduser**   添加用户
  - **userdel/deluser**   删除用户
- **man、info** : 获取帮助命令
  - **man passwd** #passwd命令的帮助手册出现在屏幕上
  - **info passwd**
  - **man read** #read系统调用的帮助手册
  - **ls --help**

Ubuntu设置root密码:  
`sudo passwd root`





# man手册

## ■ Linux **man** 手册页的内容区域

- 1 可执行程序或shell命令
- 2 系统调用
- 3 库调用
- 4 特殊文件
- 5 文件格式与约定
- 6 游戏
- 7 概览、约定及杂项
- 8 超级用户和系统管理员命令
- 9 内核例程

➤ 查看open系统调用: **man 2 open**





# 常用的几个命令

- **whatis**: 得到任何Linux命令的更短的描述命
- **whoami**: 显示用户名
- **hostname**: 显示登录上的主机的名字
- **uname**: 显示关于运行在计算机上的操作系统的信息
- **id**: 显示用户id和组id等信息

| 查·论·编      | Unix命令程序和壳层内建命令  | [隐藏] |
|------------|--|------|
| 文件系统       | cat · cd · chmod · chown · chgrp · cksum · cmp · cp · dd · du · df · file · fsck · fuser · ln · ls · mkdir · mount · mv · pax · pwd · rm · rmdir · size · split · tee · touch · type · umask               |      |
| 程序         | at · bg · chroot · cron · fg · kill · killall · nice · pgrep · pkill · ps · pstree · time · top  |      |
| 用户环境       | clear · env · exit · finger · history · id · logname · mesg · passwd · su · sudo · uptime · talk · tput · uname · w · wall · who · whoami · write  |      |
| 文本编辑       | awk · banner · basename · comm · csplit · cut · diff · dirname · ed · ex · fmt · fold · head · iconv · join · less · more · nl · paste · sed · sort · spell · strings · tail · tr · uniq · vi · wc · xargs |      |
| 壳层内建       | alias · echo · expr · printf · sleep · test · true和false · unset · wait · yes  |      |
| 网络         | dig · host · ifconfig · inetd · netcat · netstat · nslookup · ping · rdate · rlogin · route · ssh · traceroute   |      |
| 查找         | find · grep · locate · whatis · whereis  |      |
| 文档         | apropos · help · man   |      |
| 杂项         | bc · dc · cal · lp · lpr   |      |
| Unix实用程序列表 |  |      |

Linux常用命令查询: <http://zh.wikipedia.org/zh-cn/Unix实用程序列表>





# 常用命令

## ■ 列出文件名和显示工作目录

- `ls *.c`

## ■ 路径设置

- `PATH=~/.bin:$PATH:.` 搜索路径中增加~/bin和.目录

## ■ 创建和显示目录

- `pwd` print working directory, 显示工作目录
- `mkdir` make directory, 创建目录
- `rmdir` remove directory, 删除目录
- `cd` change directory, 改变当前的工作目录

## ■ 两个特殊目录

- “.”代表当前工作目录
- “..”代表当前目录的父目录





## 例：

```
$ pwd
/home/faculty/sarwar
$ ls -aC
. .. .cshrc .login ece 231 ece345 ece441 ece445 ece446 personal linuxbook
$ cd linuxbook
$ cd examples
$ pwd
/home/faculty/sarwar/linuxbook/examples
$ ls -C
chapter1 chapter2 chapter3 chapter4
$ mkdir dir1
$ ls -C
chapter1 chapter2 chapter3 chapter4 dir1
$ cd dir1
$ pwd
/home/faculty/sarwar/linuxbook/examples/dir1
$ cd ..
$ rmdir dir1
$ ls -C
chapter1 chapter2 chapter3 chapter4
$ cd ~
$ pwd
/home/faculty/sarwar
```





## 常用命令 (续)

### ■ 显示文件内容

- **cat** 同时显示一个或多个文件的内容

\$ **cat sample** 显示sample文件的内容

- **more**、**pg** 一次显示一个屏幕的内容

\$ **more sample phones** 一次显示一屏幕的sample文件的内容，然后以同样的方式显示phones文件的内容

### ■ 显示日历命令

- **cal [[month] year]**





## 常用命令（续）

### ■ 为命令创建假名（别名）

- **alias** [name[=string]...] 为“name”命令建立别名“string”
- **unalias** 删除别名

### ■ 显示系统运行时间命令 **uptime**

**\$ uptime**

11:39 AM up 7:04, 11 users, load average: 0.08, 0.12, 0.17

### ■ 清楚屏幕命令 **clear**

### ■ 改变用户的身份

- 命令语法: **su** [-][-c <command>] [username]
- 常用选项/参数:

-c < command > 执行完指定的指令后，即恢复原来的身份。

- 改变身份时，也同时变更工作目录，及HOME、SHELL、PATH变量等

username 指定要变更的用户名。若不指定此参数，则为root用户。

- 例：下面su命令将你的用户身份转换为root，当然，你必须输入root密码。

**\$ su**

**Ubuntu**隐含root账号，使用**sudo passwd root**命令设置root的密码后，才能使用**su**命令

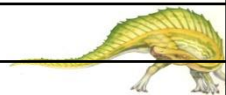






# 常用命令

| 命令           | 功能                              |
|--------------|---------------------------------|
| <Ctrl-C>     | 终止当前的命令或程序                      |
| <Ctrl-D>     | 结束输入，或退出Linux系统，或从上层shell返回     |
| <Ctrl-Z>     | 暂停当前命令执行                        |
| cp           | 拷贝文件                            |
| echo \$SHELL | 显示正在运行的shell名字                  |
| exit         | 结束当前的shell                      |
| hostname     | 显示你所登录的主机的名字                    |
| login        | 用一对正确的用户名/密码登录到计算机系统            |
| logout       | 退出登录                            |
| ls           | 显示文件和目录的信息                      |
| man          | 浏览关于一个命令或主题的帮助手册                |
| mv           | 移动或重命名文件                        |
| passwd       | 修改密码                            |
| set          | 在bash中显示和修改环境变量                 |
| uname        | 显示关于计算机正运行的操作系统的信息              |
| w            | 比who命令更加详细地列出系统上用户的信息           |
| whatis       | 显示一个命令的简要描述                     |
| whereis      | 在标准路径（非用户指定的路径）下搜索与指定命令相关文件的全路径 |
| which        | 当某个工具或程序有多个副本时，用which来识别哪个副本在运行 |
| who          | 显示现在正在使用系统的用户的信息                |





## 2.5 Linux的编辑器

---





# Linux的编辑器

## ■ 文本界面下，常用的文本编辑器有：

- **vi**编辑器：UNIX类操作系统通用的全屏幕编辑器，只要你习惯于操作，你会觉得它比任何的编辑器都好用，且功能强大。
- **vim**编辑器：vi的增强版本，是vi的克隆，是**基于GUN软件**。
- **nano**编辑器：一种风格很像Microsoft DOS的文本编辑器。一些发行版没有安装。
- **emacs**编辑器：GNU编辑器，功能强大的全屏幕编辑器。

## ■ 图形界面下，常用的文本编辑器有：

- **emacs**编辑器：编程编辑器
- **gedit**或**kedit**编辑器：全屏幕文本编辑程序

操作方便，  
建议使用

## ■ 要区分文本编辑器和排版工具不同，文本编辑器不象Openoffice、Word或WPS那样可以对字体、格式、段落等其他属性进行编排。





# vi 编辑器

- vi是Linux/Unix世界里最常用的全屏编辑器，所有的Linux系统都提供该编辑器，而Linux也提供了vi的加强版——vim，同vi是完全兼容，存放路径一般为/usr/bin/vim，vim软件及有关信息可以从[www.vim.org](http://www.vim.org)获得。
- 多数的Linux系统中vi命令是vim的别名（符号链接），你可以通过alias命令或which vi命令查看一下，所以，当您启动vi命令时，实际运行是vim程序。在本节内容中，我们不对vi和vim加以区别，统一使用vi命令。
- 在Ubuntu发行版中使用vim需要安装基础版的vim包。  
\$ `sudo apt-get install vim`





## vi 编辑器(续)

### ■ vi有两种操作方式，分别是：

- 命令模式（command mode），由击键命令序列（vi编辑器命令）组成，完成某些特定动作；
- 插入模式（insert mode），允许你输入文本。

### 图 vi文本编辑器的操作模式

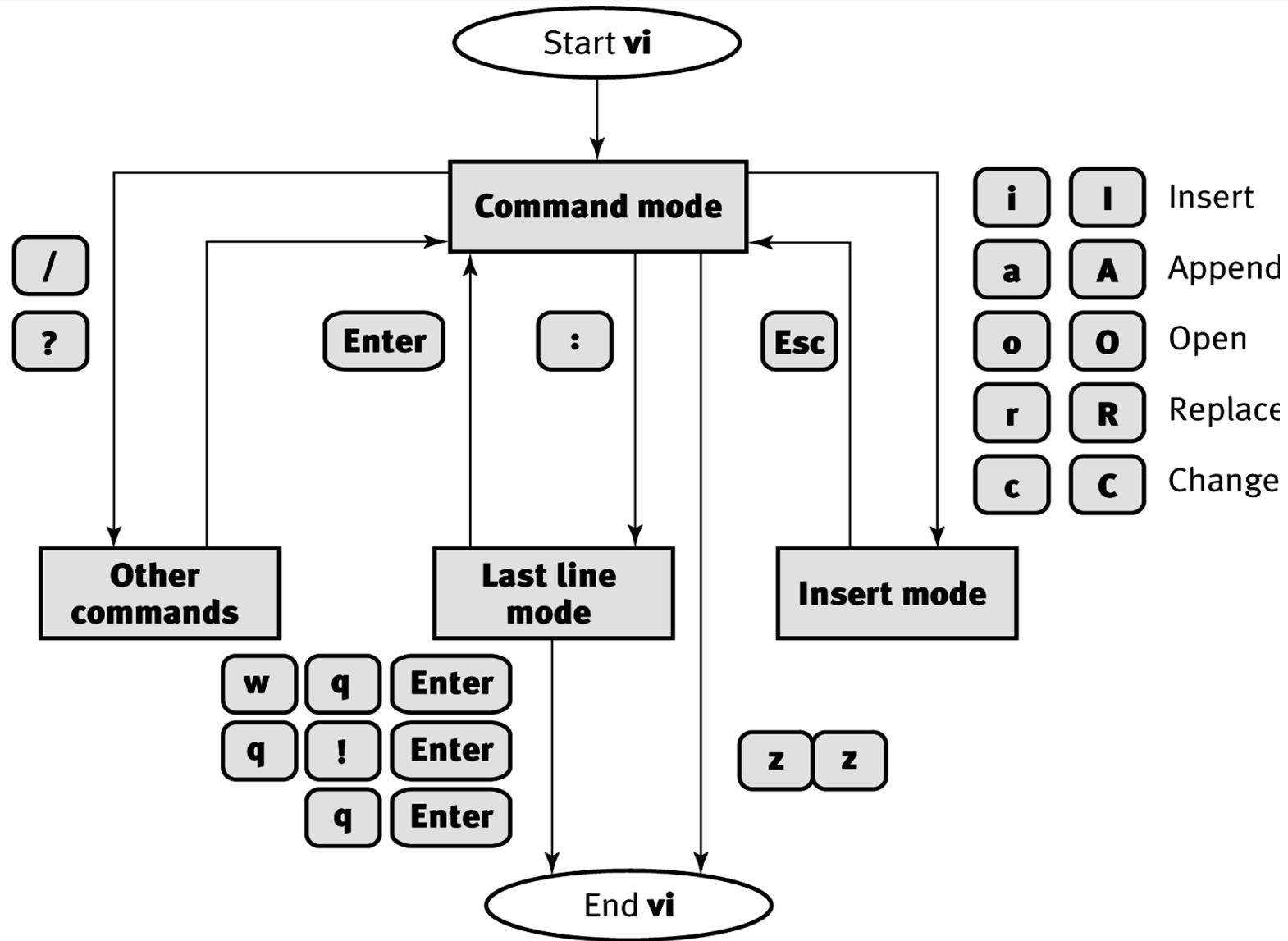
### ■ vi的进入与离开

- 在系统提示符“\$”下键入命令**vi**，后面跟上想要编辑（或者建立）的文件名，vi可以自动载入所要编辑的文件或是开启一个新文件。
- vi的退出，可以在命令模式使用命令“**:wq**”或者“**:q!**”，前者的功能是写文件并从vi中退出，后者的功能是从vi中退出，但不保存所作的修改（注意冒号）。





## vi 编辑器(续)



图vi文本编辑器的操作模式





# Vi 编辑器

version 1.1  
April 1st, 06

## vi / vim graphical cheat sheet

**Esc**  
normal  
mode

|                 |                   |                 |                  |                |               |                |                   |                 |                  |                |                 |               |
|-----------------|-------------------|-----------------|------------------|----------------|---------------|----------------|-------------------|-----------------|------------------|----------------|-----------------|---------------|
| ~ toggle case   | ! external filter | @ play macro    | # prev ident     | \$ eol         | % goto match  | ^ "soft" bol   | & repeat :s       | * next ident    | ( begin sentence | ) end sentence | "soft" bol down | + next line   |
| . goto mark     | 1                 | 2               | 3                | 4              | 5             | 6              | 7                 | 8               | 9                | 0 "hard" bol   | - prev line     | = auto format |
| Q ex mode       | W next WORD       | E end WORD      | R replace mode   | T back 'till   | Y yank line   | U undo line    | I insert at bol   | O open above    | P paste before   | { begin parag. | } end parag.    |               |
| q record macro  | w next word       | e end word      | r replace char   | t 'till        | y yank        | u undo         | i insert mode     | o open below    | p paste after    | [ misc         | ] misc          |               |
| A append at eol | S subst line      | D delete to eol | F "back" find ch | G eof/ goto ln | H screen top  | J join lines   | K help            | L screen bottom | : ex cmd line    | " reg. spec    | bol/ goto col   |               |
| a append        | s subst char      | d delete        | f find char      | g extra cmds   | h ←           | j ↓            | k ↑               | l →             | : repeat t/T/f/F | ' goto mk. bol | \ not used!     |               |
| Z quit          | X back-space      | C change to eol | V visual lines   | B prev WORD    | N prev (find) | M screen mid'l | < un-indent       | > indent        | ? find (rev.)    |                |                 |               |
| Z extra cmds    | x delete char     | c change        | v visual mode    | b prev word    | n next (find) | m set mark     | , reverse t/T/f/F | . repeat cmd    | / find           |                |                 |               |

|                 |   |
|-----------------|---|
| <b>motion</b>   | moves the cursor, or defines the range for an operator              |
| <b>command</b>  | direct action command, if <b>red</b> , it enters insert mode        |
| <b>operator</b> | requires a motion afterwards, operates between cursor & destination |
| <b>extra</b>    | special functions, requires extra input                             |
| q.              | commands with a dot need a char argument afterwards                 |

bol = beginning of line, eol = end of line,  
mk = mark, yank = copy

words: `quux(foo, bar, baz)`  
WORDS: `quux(foo, bar, baz)`

### Main command line commands ('ex'):

:w (save), :q (quit), :q! (quit w/o saving)  
:e f (open file f),  
:%s/x/y/g (replace 'x' by 'y' filewide),  
:h (help in vim), :new (new file in vim),

### Other important commands:

CTRL-R: redo (vim),  
CTRL-F/-B: page up/down,  
CTRL-E/-Y: scroll line up/down,  
CTRL-V: block-visual mode (vim only)

### Visual mode:

Move around and type operator to act on selected region (vim only)

### Notes:

- (1) use "x before a yank/paste/del command to use that register ('clipboard') (x=a..z,\*) (e.g.: "ay\$ to copy rest of line to reg 'a')
- (2) type in a number before any action to repeat it that number of times (e.g.: 2p, d2w, 5i, d4j)
- (3) duplicate operator to act on current line (dd = delete line, >> = indent line)
- (4) ZZ to save & quit, ZQ to quit w/o saving
- (5) zt: scroll cursor to top, zb: bottom, zz: center
- (6) gg: top of file (vim only), gf: open file under cursor (vim only)

For a graphical vi/vim tutorial & more tips, go to [www.viemu.com](http://www.viemu.com) - home of ViEmu, vi/vim emulation for Microsoft Visual Studio





## vi 编辑器(续)

### ■ vi的插入模式:

- 在命令模式下正确定位光标之后, 可用一下命令切换到插入模式: 表 插入模式下的重要按钮。
- 如果用户想利用已有的文件内容, 可以使用命令“: i filename”, 则vi将指定文件的内容输入当前光标的下一行, 且vi仍处于命令模式。
- 退出插入模式的方法是, 按ESC键或组合键Ctrl+l

### ■ vi的命令模式:

- 表 命令模式下的重要命令
- 表 光标移动和键盘编辑命令
- 表 复制和粘贴文本命令yank和put







## vi 编辑器(续)

- 表列出了在命令模式和插入模式下都可用的一般语法及其变形的具体示例。

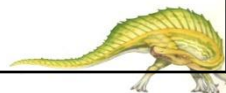
| 表     | vi命令语法示例                |
|-------|-------------------------|
| 命令    | 动作                      |
| 5dw   | 从当前光标位置开始删除5个字          |
| 7dd   | 从当前行开始删除7行              |
| 7o    | 在当前行后面开辟7个空行            |
| 7O    | 在当前行前面开辟(插入)7个空行        |
| c2b   | 修改光标前面2个字               |
| d7,14 | 将缓冲区中第7行至14行删除          |
| 1G    | 将光标置于文件首行               |
| 10yy  | 将后面10行（从当前行开始）拷贝到临时缓冲区中 |





## vi 编辑器(续)

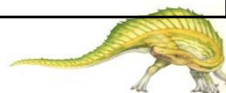
| 表   | 插入模式下的重要按键             |
|-----|------------------------|
| 按键  | 行为                     |
| <a> | 在光标所在字符后添加文本           |
| <A> | 在当前行最后一个字符后添加文本        |
| <C> | 开始修改操作，允许你更改当前行文本      |
| <C> | 修改从光标位置开始到当前行末尾范围内的内容  |
| <i> | 在光标所在字符前插入文本           |
| <I> | 在当前行开头插入文本             |
| <O> | 在当前行下方开辟一空行并将光标置于该空行行首 |
| <O> | 在当前行上方开辟一空行并将光标置于该空行行首 |
| <R> | 开始覆盖文本操作               |
| <s> | 替换单个字符                 |
| <S> | 替换整行                   |





## vi 编辑器(续)

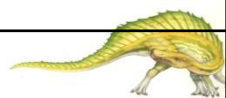
| 表            | 命令模式下的重要命令                      |
|--------------|---------------------------------|
| 命令           | 动作                              |
| d            | 删除字、行等                          |
| u            | 撤销最近一次编辑动作                      |
| p (小写)       | 在当前行后面粘贴（插入）此前被复制或剪切的行          |
| P (大写)       | 在当前行前面粘贴（插入）此前被复制或剪切的行          |
| :r filename  | 读取filename文件中的内容并将其插入在当前光标位置    |
| :q!          | 放弃缓冲区内容，并退出vi                   |
| :wq          | 保存缓冲区内容，并退出vi                   |
| :w filename  | 将当前缓冲区内容保存到filename文件中          |
| :w! filename | 用当前文本覆盖filename文件中的内容           |
| ZZ           | 退出vi，仅当文件在最后一次保存后进行了修改，才保存缓冲区内容 |





## vi 编辑器(续)

| 表        | 光标移动和键盘编辑命令            |
|----------|------------------------|
| 命令       | 动作                     |
| <1G>     | 将光标移到文件首行              |
| <G>      | 将光标移到文件末行              |
| <0>(数字0) | 将光标移到当前行首个字符           |
| <Ctrl-G> | 以行列号形式报告光标位置           |
| <\$>     | 将光标移到当前行最后一个字符         |
| <w>      | 将光标每次前移一字              |
| <b>      | 将光标每次倒退一字              |
| <x>      | 删除光标位置上的字符             |
| <dd>     | 删除当前光标所在行              |
| <u>      | 撤销最近一次所做的修改            |
| <r>      | 用随后键入的一个字符替换当前光标位置处的字符 |





## vi 编辑器(续)

| 表     | yank 和put命令的语法示例       |
|-------|------------------------|
| 命令语法  | 完成的动作                  |
| y2w   | 从当前光标位置开始向右复制2个字       |
| 4yb   | 从当前光标位置开始向左复制4个字       |
| yy或Y  | 复制当前行                  |
| p（小写） | 在当前光标位置后插入复制的文本        |
| P（大写） | 在当前光标位置前插入复制的文本        |
| 5p    | 在当前光标位置后将缓冲区中复制的文本粘贴5次 |





# vi 编辑器(续)

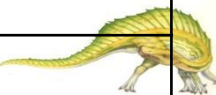
## ■ Substitute (搜索和替换), Substitute command format:

: [range]s/old\_string/new\_string[/option]

- ▶ []中的部分是可选的;
- ▶ : 是状态行命令的冒号前缀;
- ▶ range是缓冲区中有效行的范围指定 (如果省略, 当前行就是命令的作用范围);
- ▶ s 代表substitute命令;
- ▶ / 是查找的分隔符;
- ▶ old\_string 是你想要替换掉的文本;
- ▶ / 是替换的分隔符;
- ▶ new\_string 是替换上去的新文本;
- ▶ /option 是命令的修饰选项, 通常用g代表全局。

## ● Practice Session 5.5

| Table              | Subtitute命令的语法示例         |
|--------------------|--------------------------|
| 命令语法               | 完成的动作                    |
| :s/john/jane/      | 在当前行用字jane替换字john, 只替换一次 |
| :s/john/jane/g     | 在当前行用字jane替换所有的字john     |
| :1,10s/big/small/g | 在第1至第10行用字small替换所有的字big |
| :1,\$s/men/women/g | 在整个文件中用字women替换所有的字men   |





## vi 编辑器(续)

### ■ 从vi中执行shell命令：

- 命令模式下通过在命令前加:**!** 来实现。执行完一个shell命令后，vi回到它的命令模式。
- 例如：
  - ▶ 键入:**! pwd**会显示你当前目录的路径名
  - ▶ 键入:**! ls**会显示你当前目录下的所有文件名。





# emacs编辑器

- **emacs**文本编辑器可以用来编辑文本、剪辑和粘贴文本内容、提供个人日历和日记，阅读usenet新闻、发送电子邮件，同时还是一种程序语言解释器，可以编辑C、Lisp、Tev源代码文件、以及Linux的Shell。
- **emacs**是由 **Richard Stallman**发明的。是一个GNU的编辑器。**emacs**的主页为 [www.gnu.org/software/emacs/emacs.html](http://www.gnu.org/software/emacs/emacs.html)。
- 最初的**emacs**是用来编辑宏命令的，现已进一步扩充为 UNIX用户中装机用户数量最大、功能最齐全的免费文本编辑器了。







## 小结

### ■ 本章需要掌握：

- 学会安装Linux系统
- Linux的登录和退出
- 要求在文本界面和图形界面下分别掌握一种编辑器的简单使用
- 命令：man、info、su、whoami、hostname、uname、who、whatis、whereis、uptime、cal、echo、pwd、reboot等
- 知识：shell

### ■ 本章需要了解：

- 命令：cp、exit、login、ls、mv
- 知识：元字符、环境变量





# 思考题

---

- 使用Linux的终端（命令行界面）有很多多种方式，请查找相关资料依次列出，并尽可能实践之。





# End of chapter

