**Linux创始人：Linus**

**操作系统**是计算机系统的基本系统软件。操作系统是所有软件的核心。操作系统负责控制、管理计算机的所有软件、硬件资源，是惟一直接和硬件系统打交道的软件，是整个软件系统的基础部分，同时还为计算机用户提供良好的界面。因此，操作系统直接面对所有硬件、软件和用户，它是协调计算机各组成部分之间、人机之间关系的重要软件系统。

Linux系统组成

**Linux内核**(Linux Kernel)：

是Linux操作系统的核心程序，其必须完成以下内容：

管理程序的运行，为程序分配资源，并且处理程序之间的通信。

管理对文件系统的读写，把对文件系统的操作映射成对磁盘或其它块设备的操作。

管理存储器，为程序分配内存，并且管理虚拟内存。

管理输入、输出，将设备映射成设备文件。

管理网络。

**Linux Shell**：

Shell:是一个命令解释器，是内核和用户之间的接口。

Shell:是一个编程语言。(Shell script)

系统的启动过程: init -> login -> passwd -> shell

Shell提供内核的保护层

对用户屏蔽内核的复杂性

保护内核以免用户误操作造成损害

Shell在用户登陆后启动

Shell和用户对话的基本单位: Linux命令

**系统各层次的关系**

硬件是实体工作者

内核能够控制硬件工作

Shell是用户与内核沟通的桥梁，是中间站

**Linux应用程序**

标准的Linux系统都具有一整套称为应用程序的程序集，包括文本编辑器、Xwindow、办公套件、Internet工具和数据库等。

**登录**

$login: UserName

$password: [your password]

**退出**

〈Ctrl-d〉，$ exit，$ logout

$date 显示日期和时间（-s设置日期/时间）

$cal [month] [year] 显示日历

$cal 显示当前月

$cal [n] 表示年0<n<10000

$clear 清屏.

$echo 在标准输出上显示字符串

\ 换行符

$who 显示登陆到系统的所有用户的详细信息

$whoami 显示当前用户有效帐号，系统使用该帐号决定用户权限

$ mail <username> 发送E-mail

$mail 接收 E-mail

U 未读的邮件.

N 收到的新邮件.

? 或 & as the mail subsystem prompt(系统提示符)

在邮件系统提示符 ? 下可输入的命令：

d 删除信息

R 回复邮件.

q 退出mail平台，保存之前的操作

t 显示信息

enter 下一个

数字n 察看第n个邮件

一旦邮件被打开，其将被放在 $HOME/mbox目录中，若想再看这些信息则用：mail –f

# wall 广播系统管理信息

$write <username> 若一用户在线，可用write命令发送信息。

按<ctrl-d> 来结束会话.

EOF 说明另一个人结束会话,这一方也需按<ctrl-d> 来结束会话

$mesg n/y 决定用户是否接收来自write和 wall 命令发送的消息，但对root用户的发送的消息不起作用.

<backspace> 删除

<ctrl-c> 中断当前的命令并返回Shell.

<Ctrl-d> 中断当前的通信或从文件中退出。

<ctrl-u> 删除整行.

命令是大小写敏感的，不同于Windows.

命令、选项和参数之间必须用空格格开

$man [命令] 是一个帮助命令,可以通过这个命令显示需要命令信息。

<Space bar> :每次向后翻一页

<Enter> :每次向后移动一行

<b> 　:　每次向前翻一页

<f> 　:　每次向后翻一页

<ctrl-c> or q: 退出man 命令.

{ } :强制选项

[ ] :[ ] 内的内容为可选项

**文件**是Linux用来存储信息的基本结构， 它是被命名（称为文件名）的、存储在某种媒介（如磁盘、光盘和磁带等）上的一组信息的集合。

**文件(file)的命名规则**

Linux文件名的最大长度为256个字符

通常由字母、数字、“.”（点号）、“\_”（下划线）或“-”（减号）组成，文件名中不能含有“/”符号。

避免使用具有特别意义的字符： ?,\*,@, #,$,&,(,),\,|,;,‘,“,`,<,>,[,]等

隐藏文件一般是以“.”符号开头；

在文件名中的空格或制表符，在引用文件时必须用引号将其括起来；

避免使用 ‘+’和‘-‘符号作为文件名的第一个字符

大小写敏感

为了便于管理和识别，也可以把扩展名作为文件名的一部分。圆点用于区分文件名和扩展名。

**文件类型**

**普通文件**（regular file）二进制文件，文本文件

**二进制文件**

以二进制形式存储在电脑中

用户一般只有通过相应的软件才能将其显示出来

一般是可执行程序、图形、图像、声音等

**文本(txt)文件**

以文本的ASCII码（美国信息互换标准代码）形式存储在电脑中，其内容都是可读字符。

以“行”为基本结构的信息组织和存储方式

**目录文件**（directory）是一个包含文件的容器，用于存放目录中文件列表信息。

目录文件存储一组相关文件的位置、大小等与文件有关的信息，但它不包含具体的文件内容，因而和常规文件相比占用的磁盘空间很小。

每一项(entry) 主要表示的是一个文件名(或子目录名)以及文件的**索引节点号**(i-node number ) 。

**索引节点号：**指向该文件内容所在的数据块的位置，记录了该文件的属性

通过 i-node表与文件之间建立对应关系，目录文件是存放文件名唯一的地方，但文件名不放在 i-node表中.

访问一个文件或子目录时，首先访问他所在的目录，找到该文件的i-node值，然后再查找i-node 表找出相应的i-node项（i-node entry），从而找到了相应的数据。

**设备文件**(device file)Linux中，设备被当成文件来处理。

**块设备**：以块为单位进行随机存取。常见块设备：软盘、光盘、硬盘。

**字符设备**：以单个字符为单位进行顺序存取。常见的字符设备：打印机、终端、键盘、鼠标

**文件系统**指文件存在的物理空间

由一组普通文件、目录文件、设备文件和符号链接组成

树状层次结构：Linux 将分属不同分区的、单独的文件系统整理形成一个系统的，总的目录层次结构

Linux下的所有文件都依附在根目录/下

Linux 通过挂接（mount）一个文件系统将该新文件系统加入它的文件系统树中。所有的文件系统, 不管是什么类型，都挂接在文件系统树的一个目录上并且该文件系统之上的文件将掩盖掉这个挂接目录中原来存在的内容。这个目录称为挂接目录或挂接点。当文件系统被卸掉之后，挂接目录中原来的文件才再次可见。

**用户主目录（用户的登录目录或起始目录）**

每次登录后自动位于其主目录下

户主目录由系统管理员在创建帐号时建立，每个合法的用户在文件系统中都有一个唯一的起始目录。默认的主目录位于/home目录下，以该用户名命名。

用户的主目录属于用户，用户可以在此目录下进行各种文件操作

$ cd： 命令可从任何位置返回主目录

$ echo $HOME查看自己的主目录

**当前目录:**用户当前所处的工作目录

$ pwd(print working directory) 显示当前工作目录

**路径**

每个文件都有一个唯一的路径名。路径名表明了文件的位置。

绝对路径: 指出从根目录到此文件的路径，一般来说由/符号开始。

相对路径：指明从当前目录到此文件的路径

**“.”**，代表当前目录， **“..”**，代表当前目录的父目录

$ cat 将文件的文本内容一次全部显示在屏幕上

-n :打印行号

-b :打印行号,空行不编号

使用重定向进行文件的合并、建立、覆盖和添加内容等操作

$ more 可以分屏显示文件内容

f键或<Space>: 进入下一页。

b键: 前一页

<Enter>: 向下移动一行

q键或<Ctrl+C>:退出。

$ wc [ -options] [ Filename ] 统计文件中的字符数、单词数和行数。

-c：显示字符数。

-l：显示行数。

-w：显示单词数。

若不加任何选项，则会将文件中字符数、单词数和行数全部列出。输出列总按下述顺序显示:行数 单词数 字符数 文件名

若不给出文件名，则从标准输入中读取。

$ ls [-options] [filelist]:显示该目录或路径下所包含的文件及目录列表，以及文件的 相关信息。

如果不指定路径名，则列出当前目录中的所有文件

不带任何选项的ls命令只列出文件名

-a 用于列出目录中的所有文件，包括文件名以“.”开头的隐藏文件

-l 以长格式列出文件的详细信息：文件的类型、操作权限、链接数、属主名、属组名、字节数以及最近修改时间、文件名

每行的第一个字符表明文件的类型：

d 目录

- 普通文件

l 符号链接文件

b 块设备文件

c 字符设备文件

-d 显示目录名而不显示其中的文件。

-R 递归列出子目录

-r 以逆序显示文件名列表

-x 显示时以字母顺序

--color 用不同颜色区分文件类型

绿色--->可执行文件  
 蓝色--->目录  
 红色--->压缩文件  
 浅蓝色->连结文件  
 灰色--->一般文件(没有定义的文件)

颜色自定义可修改/etc/DIR\_COLORS

注：正常的显示顺序为按ASCII排序方式。

$ cd [directory]改变当前工作目录

cd ~ 与cd意义相同；

cd .. 到上一级目录，也即父目录。

$ touch [-t] [YY][MMDDhhmm] filename以当前时间创建文件或更新已有文件的时间戳

若所指定文件不存在，则自动建立指定文件(大小为零)

若要输入时间格式，至少要输入MMDDhhmm

如果没有输入时间，则默认为当前时间

$ cp [-options] src\_file dst\_file

-i：在覆盖文件之前提示用户，由用户确认。

R或-r：递归复制目录，即复制相应的目录及其所有子 目录

若dst\_file已存在，将覆盖dst\_file的原有内容

一次拷贝多个文件

$ mv [-options] src\_file dst\_file用于对文件进行名称更改或路径迁移

-f 强制执行。直接覆盖已存在的目的文件，不显示覆盖前的询问讯息。

-i 交互执行。当已存在同名的目标文件名时，在覆盖之前给出提示，由用户确认后才予以覆盖。

$ rm [-options] files删除文件或目录

-f：强制执行。

-i：交互执行，在执行删除前提示确认。

-R或r：递归的删除目录

当要删除多个文件时，多个文件名之间要用空格隔开

$ mkdir [-options] DIRECTORY创建一个或多个子目录

-p：parent的缩写。自动建立所需路径中不存在的上层目录。

-m：建立指定权限的目录。

每个新创建的目录都包含两个标准项 . (dot) 和 .. (dot-dot).

创建新的目录时，对其父目录必须有写权限。

$ rmdir 删除空目录

对该目录的父目录必须有写权限。

被删除的目录是个空目录，否则只能尝试-p选项

-p：多层次的空目录删除。由指定目录的最底层开始，逐层尝试删除空目录，当碰到非 空目录时便停止删除的动作。

$ whereis [-options] [file]显示指令的二进制码、原始码与在线手册说明文件的存放目 录

-b：只查找二进制文件；  
 -m：只查找在线手册manual 路径下的文件；  
 -s：只查找原始码文件；

whereis找不到某个文件的部分原因可能是这个文件没有存在于任何whereis命令搜索的子目录中。whereis命令检索的子目录是固定编写在它的程序中的

$ whereis -b passwd

passwd: /usr/bin/passwd /etc/passwd

$ locate [file]使用的是一个文件名数据库

查找文件位置又比whereis命令速度更快

数据库的更新是每星期执行一次,可执行 updatedb 手工更新数据库

**链接文件:**目录中每一对文件名称和索引节点号称为一个链接

**硬链接（ Hard Link）**

原文件名和链接文件名都指向相同的物理位址

如果删除硬链接文件的源文件，硬链接文件仍然存在，而且保留了原有的内容

不允许普通用户对目录建立硬链接；硬链接不能跨越文件系统（不能跨越不同的分区）

**符号链接（ Symbolic Link）**

内容是它所链接的文件的路径名

每个链接文件都有各自的索引节点号。

如果删除符号链接文件的源文件，符号链接文件将无法找到原有资料。

可用于链接目录，且符号链接能够跨越文件系统

$ ln [-s] target [link name]对一个已经存在的文件再创建一个新的链接，而不复制文 件的内容。

-s：符号链接。不加此选项代表硬连接。

Target:链接所对应的源文件

Link name：链接文件名。如果是符号链接，也可以是目录名。

ln后的链接文件名与物理文件具有相同的权限

多用户、多任务的操作系统决定了常常会有多人同时使用这部主机工作

保证每个人的隐私权-->**文件所有者**

资源共享，团队开发-->**文件所属用户组**

Linux中每个用户都至少属于一个用户组，以保证对一个用户组中的所有用户进行集中管理

**用户类型**

**文件所有者**：能够设定同组用户和其他用户对该文件的访问权限。一般来说是文件的创建者。

**同组用户**：具有相同性质的所有用户,被系统管理员分在同一组。文件所有者或系统管理员 可将文件的权限赋予组内的其它用户。

**其它用户**：文件所有者或系统管理员还可以将文件的访问权赋予系统中所有其它的用户。这 样,系统中每一位用户可能都能访问你的某一文件或目录。

**用户类型和权限的表示方法**

u 文件的所有者

g 同组用户

o　　 其它用户

a 　all=ugo,即所有用户

三组依次代表文件所有者、同组用户和其他用户对该文件的权限

读(r)：4 写(w)：2 执行(x)：1 禁止(-)：0

在移动文件时不需要被移动文件的权限，但需其所在目录具有写权限。

在目录下增删文件、子目录时要有目录的写权限（touch,rm,mkdir,rmdir）

用ls列目录要有目录的读权限

进入目录或将该目录作路径分量时(cd)要有目录的执行权限，故要使用任一个文件，必须有该文件及找到该文件的路径上所有目录分量的执行权限。

仅当要打开一个文件时，即执行涉及到文件内容的操作时，才需要文件的许可。

**修改权限**

$ chmod [-options] xyz [filelist]

$ chmod [-options] u/g/o/a +/-/= r/w/x [filelist]

-c：只有在文件权限确实改变时才进行详细说明；

-f：不打印“权限不能改变”之类的文件错误信息；

-R：递归改变目录及其内容的权限；

-v：详细说明权限的变化。

$ chown [-options] user[：group] filelist 用于改变文件的所有者,可以同时也改变其 所属用户组

仅用于系统管理员对文件和用户的管理

-R：递归改变目录及其内容的所属用户或用户组；

用户修改密码，是通过运行命令passwd来实现的，最终必须要修改/etc/passwd文件，而passwd的文件只对于root用户是可写的，而对于所有的他用户来说都是没有写权限的。SUID/SGID

**进程**

**概念**：进程是程序的执行的过程，是执行的程序

操作系统通过进程来控制对CPU和其他系统资源的访问，并且使用进程来决定在CPU上运行哪个程序、运行多久

Linux系统可以同时启动多个进程

程序是静态的，是保存在磁盘上的可执行代码和数据的集合；

进程是动态的，是Linux系统的基本调度单位

父进程和子进程

**PID**：Process Identity number。一个PID唯一地标识一个进程。

**PPID**：Parent Process ID。进程的父进程号。

**Init进程**

内核启动后的第一个进程，其PID=1

扮演终结父进程的角色，所有进程追溯其祖先最终都会落到init进程身上。

因为init进程永远不会被终止，所以系统总是可以确信它的存在，并在必要的时候以它为参照。如果某个进程在它衍生出来的全部子进程结束之前被终止，此时那些失去了父进程的子进程就都会以init作为它们的父进程。

$ ps [-options] 列出当前进程清单

确定有哪些进程正在运行以及运行的状态、进程是否结束、进程有没有僵死、哪些进程占用了过多的资源等等。

静态列表:启动这个命令时正在运行的进程的快照

不跟任何选项表示查看系统中属于自己的进程

-e：显示所有进程。

-f：全格式(full list)。

-l：长格式。

-w：宽输出。

-a：显示所有用户进程。

-u：打印用户格式，显示用户名和进程起止时间。

-x：显示与终端无关的所有进程。

-ef：察看当前系统中运行的所有进程

-aux：显示系统中所有用户进程及其所有者，并显示详细的进程信息

$ top [-] [d delay] [q] [C] [c] [S] [s] [i] [n] 列出进程清单

显示的是可根据真实情况 , 定时更新的运行进程列表

与终端设备进行交互，根据用户从终端输入的交互命令来显示或管理进程

d delay：指定更新的时间间隔(delay)，以秒计算。  
 n：指定更新的次数，到达指定次数后将会退出top。

**交互命令**

[Space] 立即刷新显示。

d 设置刷新进程的时间间隔，你会被提示输入一个数（秒）

q 退出top命令。

k 杀死某进程。你会被提示输入进程 ID 以及要发

送给它的信号。

h 显示帮助屏幕

n 显示的进程数量。你会被提示输入数量。

u 选择用户。

M 按内存用量排序。

P 按 CPU 用量排序。

**启动方式**：//////////////

手工启动：当时设置当时启动；

前台启动:启动一个进程后，如果不中断或挂起该进程，用户会一直被禁止与 Shell 进行交互，直至该进程执行结束。

经常用于一般Linux命令的执行，和一些不太耗时的进程的启动。

后台启动:启动进程后，用户仍可以与Shell进行交互。

可用于一些耗时长的作业运行。

调度启动：用户可事先进行设置安排，指定任务运行的时间或场合，到时候由系统自动启动进程完成此项任务。

前台进程:$cat file1

停止: <Ctrl+c> 挂起：<Ctrl+z>

后台进程: $command &

终止和挂起后台进程：使用kill命令

$ jobs [-options]用于显示和控制后台正在执行的和被挂起的任务序列。

-l：显示后台任务的进程号与信息。

-p：只显示后台任务的 PID。

-n：显示上次通知用户后，执行状态有更动的后台任务状态。

-r：显示执行中的后台任务。

-s：显示暂停执行的后台任务。

$ kill [-option] [signal] {PID | %job ID}向正在执行的进程发送信号

signal - 信号。与-s配合使用。不指定则送出信号 15(TERM）。

pid - 要终止的进程号。

job ID - 要终止的进程对应的后台任务编号。

-s： 指定需要送出的信号，既可以是信号名也可以是对应数字。

–l: 查询kill可以送出的信号列表

15（SIGTERM）:终止进程运行

为kill命令的缺省信号

经常用于后台进程的终止。

当某个进程占用的CPU时间过多，或是某进程已经挂起，可以用这种方法终止其执行。

除root用户外，一般用户只能终止属于自己的进程。

09(SIGKILL):强制终止进程

01:挂起(hangup) <Ctrl + z>

02:中断(interrupt)

进程挂起:前台进程的挂起：<Ctrl+z>

后台进程的挂起：kill –s 01 {PID|%job ID}

进程恢复:恢复到前台：fg %jobID

恢复到后台：bg %jobID

用户退出系统时，一般来说会结束该用户的所有正在运行的程序

nohup {command|script} & 进程在用户退出后仍能继续执行 {a|b}：a和b中必选一项

command：命令

script：程序或脚本

nohup.out文件：存放运行后的所有错误和输出信息。

**vi类型编辑器的优点**

“简单工具，多样组合”

小，避免了功能蔓延。

比 Emacs 快

可运行于任何实现了C 标准库的系统之上

更普及

**Emacs 的优点**

提供了比vi更多的功能

移植最广泛的计算机程序之一

可扩展和可定制

**Vi 编辑器**

系统必备工具：Linux系统的第一个全屏幕交互式编辑程序

功能强大：可以执行输入、删除、查找、替换等众多文本操作,但它并不是一个排版程序， 它只是一个文本编辑程序。

$ vi [-options] [+n] [file]

-r 恢复系统突然崩溃时正在编辑的文件。

-R 以只读方式打开文件

+[n] 指明进入vi后直接位于文件的第n行，如果只有“+”而不指定n，，则光标位于文本的最后一行

如果该文件不存在，会自动建立新文件。

**命令模式**

该模式下从键盘上输入的任何字符都被当做编辑命令来解释

i (当前光标位置插入)

a (光标后一个位置插入)

o (光标下一行插入新的行) esc

**输入模式**

在该模式下，用户输入的任何字符都被Vi当做文件内容保存起来，并将其显示在屏幕上。

**底行模式**

多数文件管理命令都是在此模式下执行的（如文件保存、退出vi等）

执行完命令后，自动回到命令模式。

**移动光标**

**命令模式下:**

键盘的上、下、左、右方向键

nG 　　 跳到第n行

G 跳到最后一行

0 跳到行首

$ 跳到行尾

ctrl+b 前移一页

ctrl+f 后移一页

**底行模式下**

:n 将光标移到第n行

数值--行号；

. -- 光标所在行的行号；

$ -- 正文最后一行的行号；

例:.+5 　 将光标移动到当前行往下的第5行。

:345 将光标移到第345行

**字符串搜索**

**命令模式**

正向搜索（/） 反向搜索（?）

n 继续搜索，找出 str1 字符串下次出现的位置

N 反方向搜索

**字符串替换**

**底行模式**

[n1, n2]s/str1/str2/[g][c]

在第n1行到n2行的范围内将字符串str1用str2代替:

c 每次替换都由用户确认

g 对行中搜索字符串的每次出现进行替换。

不加g 只对行中搜索字符串的首次出现进行替换；

**撤销和显示当前状态**

**命令模式**

U 撤销上一命令结果

Ctrl-g 命令显示当前编辑文本的状态，包括文本共有多少行、文件名以及目前光标停在多少行。

**保存和退出**

**底行模式**

保存：

:w 编辑的内容写入原始文件，用来保存编辑的中间结果

:w file 将编辑的内容写入 file 文件

退出vi：

:q 在未作修改的情况下退出

:q! 放弃所有修改，强制退出编辑程序

:wq 将编辑的内容写入原始文件并退出编辑程序

**在vi内使用Shell**

**底行模式**

用户不用退出vi就可以运行任何Linux命令.

:! command

执行完后,再按一下<Enter>键回到命令方式

**Shell脚本**

把一组Shell命令或Shell程序语句组合在一起，放在一个文件 (script)中

当要执行这组命令或程序语句时，可以直接执行这个文件 (script)

称这个文件为:命令文件/Shell程序/脚本

**Shell的种类**

Bsh(Bourne Shell) UNIX 的缺省Shell,Shell编程能力很强,但在处理与用户的交互方面做得不如其他Shell。

Csh(C Shell)它能提供Bourne Shell所不能处理的用户交互特征，如命令补全、命令别名、历史命令替换等, 但C Shell的编程能力不如Bourne Shell。但它的语法和C语言类似，所以C程序员将发现C Shell很顺手。

Ksh(Korn Shell)　融合了C Shell和Bourne Shell的优点，并和Bourne 完全兼容。Korn Shell的效率很高，其命令交互界面和编程交互界面都很不错。

Bash(Bourne Again Shell) Linux的缺省Shell, Bsh的扩展，在Bsh基础上增加、增强了许多特性。提供命令补全、命令编辑、命令历史表等功能，具有Csh和Ksh的优点

TC Shell： Csh的扩展版

PdkShell：K Shell的延伸

$Shellname:启动一个指定类型的子Shell

$exit/<Ctrl+D>:退出子Shell

**Shell环境变量**：

保存路径名、文件名、数字等信息

用户定制用户自己的工作环境

保存有用信息，使系统获知用户相关设置

保存暂时信息

**定义:**变量名= 值 或 {变量名= 值} 花括号能防止Shell误解变量值

$ echo $aaa **显示**环境变量：显示单个环境变量取值.

$ unset 变量名 **清除**环境变量

$ set 显示所有Shell环境变量

$ env 显示所有当前用户环境变量

**种类**

本地环境变量：

只能在当前Shell中使用的变量，不传递给该Shell创建的任何子进程

优点：用户不能设置、修改其他的Shell或进程的本地环境变量

全局环境变量：

可以用于所有进程的变量

传统上，所有全局环境变量均为大写

**profile**

设置途径：

/etc/profile：存放系统管理员设置的环境变量

~/.bash\_profile

用户主目录下的.profile(.bash\_profile)：用于增加新变量或覆盖/etc/profile的设置

每次登陆时这些设置值都会被初始化

在命令行设置：用户注销时这些值将丢失

$ export

显式地给环境变量赋空值

Variablename=

Variablename=“”

Variablename=‘’

环境变量名只能是字母、数字、下划线的组合，但是数字不能是开头字符。

等号两边不能直接接单独的空格符。

如果变量所取的值中包含空格，必须使用引号将变量取值括起来。

必要时需要用‘\ ’来将特殊符号（如Enter, $, \, 空格, 引号）变成一般符号；

变量名为大小写均可。通常：大写字母为系统预设的变量，小写字母用于用户自定义的变量

$ read variables读入变量值

variables可以是多个变量，它们之间用空格隔开

若读入的内容个数比变量多，则将多出的都赋给最后一个变量

若变量比读入的内容个数多，则多出的变量将设为空值

$ $variable或${variable}用已赋给该变量的值来替换

**嵌入Shell变量**

HOME变量：保存用户主目录的完全路径名

LOGNAME变量：保存登陆用户名

PS1变量：保存主提示符

基本提示符包含Shell提示符，缺省对超级用户为#，其他为$。可以使用任何符号作提示符.

PATH变量：保存进行命令或脚本查找的目录顺序，不同的目录路径名之间用冒号分隔开。 PATH=/bin:usr/bin:/etc:/home/team01/bin:.

加入新的查找路径：$PATH=$PATH:filelist

**Shell特殊字符**

**单引号**括起来的所有字符都作为普通字符出现

**双引号** $/\/引号有特殊含义

**反引号**括起来的内容被Shell解释为命令行，执行时，Shell首先执行该命令行，并用它的 标准输出结果取代整个反引号（包括反引号括起来的部分）。

反引号的嵌套使用：内层的反引号要使用反斜线将其转义

**反斜线**如果下一个字符有特殊含义，反斜线屏蔽其特殊含义。& \* + ^ $ ` " | ?

**通配符**

**\*** 可以匹配文件名中的任何字符串

**？**可以匹配文件名中的任何单个字符

**[…]** 匹配方括号[ ]中指定范围内的字符，可以使用横杠-来连接两个字母或数字，以表示范围

**[!...]** 匹配不在指定范围内的任何字符

**注释符号 #**

**位置变量参数 $0,$1,$2,…**

Shell在解释用户命令时,将把命令行的第一个字作为命令,而其它字作为参数通过位置变量传递给程序

第九个以后的参数都不会被访问

位置变量$0指命令对应的可执行名

**特定变量参数** 用于传递Shell程序运行时的相关控制信息

$# 传递到脚本的参数个数

$? 最后一个程序或命令的执行状态，用数字表示：0表示执行没有错误，其他任何值表明有错误。

$\* 所有参数

$$ 脚本运行的当前shell的PID

$@ 所有参数，每个参数都用双引号括起

进程间变量的关系：默认各自独立

环境变量的值只能传递给子进程，而不能传递给它所在的父进程

**Shell脚本的运行方式**

**sh** bash启动了一个子Shell,Shell命令都是在子Shell中执行, Shell命令执行完后，退回到命令提示符下时，子Shell结束

**.** 在当前Shell下执行Shell脚本

**赋予执行权限** 用户必须对该文件具有以下权限：执行权限、读权限

这种执行方式，Shell首先查找 PATH 变量中的各路径中是否有相应程序，若没有则会出错。因而需要在PATH变量中添加脚本所在的路径。这种方式也是在子Shell中执行脚本

**标准文件和文件描述符**

在Shell中执行命令的时候，每个进程都和三个打开的文件相联系，并使用文件描述符来引用这些文件

输入文件 —　标准输入 – stdin 0

命令的输入，缺省是键盘，也可以是文件或其他命令的输出

输出文件 —　标准输出 – stdout 1

命令的输出，缺省是屏幕，也可以是文件

错误输出文件 —标准错误输出 – stderr 2

命令产生错误的输出，缺省是屏幕，同样也可以是文件

**输入输出重定向**

$ Command operator file

输入重定向 < ,输出重定向 >

**错误重定向**

2>

/dev/null 是一个特殊的文件，它总是一个空文件

Command > outfile 2> errfile < infile

以infile为标准输入，将标准输出送到outfile,标准错误输出送到errfile.

Command >> outfile 2>> errfile < infile

以infile为标准输入，将标准输出追加到outfile,标准错误输出追加到errfile.

把stdout和stderr同时重定向到outfile

$Command > outfile 2>&1

**管道**把一个命令在屏幕上的输出传递给另一个命令作为输入

$ 命令1 |命令2

**过滤和正则表达式**

正则表达式是一些字符串模式的集合，由一些特殊字符或进行模式匹配操作时使用的元字符组成。也可以使用规则字符作为模式中的一部分

以行为单位，来进行字符串的处理行为

符号 含义

^ 只匹配行首，用在字符串前面

$ 只匹配行尾，用在字符串后面

X\* 0个或多个字符X

. 只匹配任意单字符

[字符表] 字符表中的任意字符

[^字符表] 任意不在字符表中的字符

\ 屏蔽其后的特殊字符的特殊含义

\{n\} 前导的正则表达式重复n次

\{min,max\} 前导的正则表达式重复min~max次

$ grep [-option] pattern files允许对文本文件的内容进行模式查找。如果找到匹配模式，grep显示包含该模式的所有行。

-c 只输出匹配行的计数。

-n 显示匹配行及行号。

-v 显示不包含匹配文本的所有行。

pattern代表一个匹配模式,若匹配模式中包含字符串参数，则最好用引号括起，以免引起歧义

files指查找针对的文件。若指定了多个文件名，结果中会在每一行的前面显示文件名

**find 命令** 遍历当前目录甚至于整个文件系统来查找某些文件或目录

只要你具有相应的权限，即使系统中含有网络文件系统(NFS)，find命令在该文件系统中同样有效

倾向于把它放在后台执行

$ find path expression [-exec -ok] [command]

path:所查找的目录路径，对目录的搜索是递归的

expression:希望查询的文件的匹配标准或说明

名称标准

-name file寻找名为file的文件，要找的文件名包括在引号中,可以使用通配符。

-user name按照文件属主来查找文件

-group name按照文件所属的组来查找文件

-type type查找某一类型的文件(b、c、d、f、l)

-size n[c] 查找文件长度为n块(block)的文件，带有c时表示文件长度以字节计

时间标准

-mtime n按照文件的更改时间来查找文件：

-n最后的更改时间在n天以内。

n , +n 更改时间距现在n天以上。

-exec command：find命令对匹配的文件执行该参数所给出的Shell命令command。

-ok command：和-exec的作用相同，只是在执行每一个命令之前，都会给出提示，让用户来确定是否执行。

command {} \;

多个命令在一行输入时，用 “;”作为命令间分隔符。

**退出状态**

每个程序或命令执行完后，都会给系统返回一个退出状态，指示是否成功运行

退出状态用数值表示：

成功执行：0

否则：非零值(说明执行失败或某处有错误)

特定变量参数$?

保存最后程序的退出状态值

检验退出状态命令：

$echo $?

$ exit [n]立即终止Shell程序的执行

n为一数字，表示希望返回的退出状态。如果不指定，就采用exit之前最后所执行命令的退出状态

如果直接在终端上执行exit命令，会导致从系统注销，因为它会终止登陆Shell的执行。

&&和||

**算术表达式的计算**

**$ $((expression))**Shell的算术扩展提供变量的整数运算机制

expression：算术表达式，由变量/常数和运算符组成，运算符源于c语言。

expression内可以使用括号来强制分组

在$(( expression ))的双小括号内，空格可以任意添加

表达式内的变量若未定义，则当作其值为0

算术扩展可以用来判断真假

echo $((expression))

variable=$((expression))

**$ expr 表达式** 进行整数运算

表达式由操作数和运算符组成。操作数一般是整数，也可以是字符串。

表达式的各部分(操作数和运算符之间)必须以空格分隔。

表达式中某些对shell有特殊意义的字符必须转义。

变量名前加$

可以使用反引号改变计算次序

可用于字符串测试

+ 加法

- 减法

\\* 乘法

/ 整除

% 求模(余数)

=

**$ test expression /$ [ expression ]** 用于测试一种或几种条件

使用方括号时，要注意在条件两边加上空格。涉及到变量值（$variable）的比较时最好把变量取值用双引号括起来，以免Shell误解变量值

测试结果反映在退出状态中,而不是直接显示输出：

条件为真 =〉退出状态为0

条件为假 =〉退出状态非0

**字符串操作符**

string1 = string2 字符串相等

string1 != string2 字符串不等

string 字符串非空

等号和不等号操作符两边必须有空格

涉及到变量值比较时最好把变量值（$variable）用双引号括起来

根据长度、内容判断字符串是否相等

**整数操作符**

int1 –eq int2 数值相等

int1 –ne int2 数值不等

int1 –lt int2 int1< int2

int1 –gt int2 int1>int2

int1 –le int2 int1<= int2

int1 –ge int2 int1>=int2

**文件操作符** 测试文件状态

-e file 文件file存在

-d file 文件file是一个目录

-f file 文件file是一个普通文件

-s file 文件file大小不为0

-r file 可读（对文件所有者的权限）

-w file 可写

-x file 可执行

**逻辑操作符** 测试多个条件的与(and)或(or)组合

-a 逻辑与

-o 逻辑或

! 逻辑非