第二章　Linux的使用

2.1 安装 Red Hat Linux

Red Hat 是目前最流行的 Linux 操作系统，也是我们的实验系统的平台，因此将以 Red Hat Linux 7.2 为例介绍其安装过程。Red Hat Linux 的安装方法很多，我们采用的是从 CD-ROM 安装。其安装光盘共有两张，第一张可直接从光盘启动，包含大部分的软件包和一些安装工具，第二张光盘则包含许多附加的软件包。下面是安装过程和注意事项：

一．启动安装程序

用 Linux 的第一张光盘，从光驱引导启动系统。进入一个启动界面，显示“boot:”提示符，直接回车（enter），选择图形模式进行安装。

二．选择使用的语言

三．选择默认的键盘设置

四．选择默认的鼠标设置

五．设置安装类型

Red Hat Linux 提供了个人桌面、工作站、服务器和定制等多种安装类型，根据具体情况选择个人桌面或定制方式。

六．进行硬盘分区

对硬盘进行分区是一件非常危险的工作，若没有必要的把握，最好先对重要的数据进行备份，以防不测。对于 IDE 硬盘，Red Hat Linux 的驱动器标识符为“hdx~”,其中“hd”表明分区所在设备的类型，这里是指 IDE 硬盘。“x”为盘号（a 为基本盘，b 为基本从属盘，c 为辅助主盘，d 为辅助从属盘）,“~”代表分区，前四个分区用数字 1 到 4 表示，它们是主分区或扩展分区，从 5 开始就是逻辑分区。

对用户而言无论有几个分区，分给哪个目录使用，它归根结底就只有一个根目录，一个独立且唯一的文件结构。Red Hat Linux 采用了“装载”的处理方法，将一个分区和一个目录联系起来，因此每个分区都是用来组成整个文件系统的一部分。

Linux 最少需要两个分区，一个 Linux native（文件）分区，一个 Linux swap（交换）分区。其中 Linux native 分区是存放 Linux 系统文件的分区，它只能用 EXT2 的分区类型，在分区时应该将载入点设置为“/”目录。SWAP 分区则用做交换空间，它主要是把主内存上暂时不用得数据存起来，在需要的时候再调进内存内。一般建议分区方案如下：

SWAP 分区：SWAP 分区至少要等于系统上实际内存的容量，一般来说它的大小是内存的两倍。

/boot 分区，它包含了操作系统的内核和在启动系统过程中所要用到的文件，建这个分区是很有必要的，因为目前大多数的 PC 机要受到 BIOS 的限制,况且如果有了一个单独的/boot 启动分区，即使主要的根分区出现了问题，计算机依然能够启动。这个分区的大小约在 50MB—100MB 之间。

/分区：这是根目录挂载的位置。系统运行所需要的其它文件都在该分区上，这个分区的大小约在 1.7GB—5GB 之间。

假如是初次安装 Linux 系统，你最好选择自动分区的方式。当然，如果你对 Linux 较为熟悉，也可以用系统配制的硬盘管理工具 Disk Druid 来定制所需要的分区，它既可以根据用户的要求创建和删除硬盘分区，还可以为每个分区管理载入点。

七．设置文件系统为 EXT2

八．配置引导装载程序

选择 LILO 作为引导安装程序。LILO 可以安装在：第一硬盘的主引导区（MBR）或 Linux 分区的引导扇。如果你想使用 LILO 来做双启动的话，你需要选择第一种，如果是想用 Linux 启动软盘或其他系统引导器引导 Linux，请选择第二种方式，即将 LILO 安装在 Linux 分区的引导扇区。

九．网络配置

十．防火墙配置

十一．选择其它支持语言

十二．时区配置

十三．设置 root 密码

十四．选择软件包组

十五．筹建引导盘

十六．配置显卡

十七．进行安装

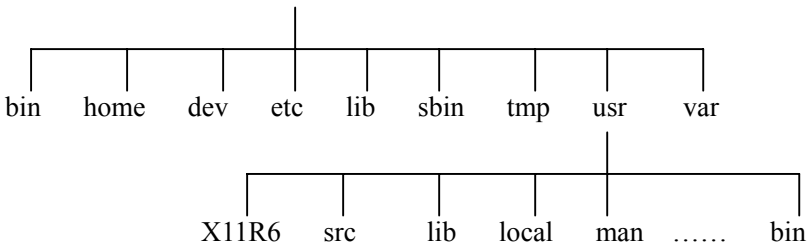
2.2 Linux 文件系统结构

2.2.1 Linux 目录结构

Linux 文件系统是树状的结构，系统中每个分区都是一个文件系统，都有自己的目录层次。Linux 会将这些分属不同分区的、单独的文件系统按树型的方式形成一个系统的总的目录层次结构。目录提供了一个管理文件方便而有效的途径，最上层是根目录，其他的所有目录都是从根目录出发而生成的。微软的 DOS 和 Windows 也是采用树型结构，但是在 DOS 和 Windows 中这样的树型结构的根是磁盘分区的盘符，有几个分区就有几个树型结构，它们之间的关系是并列的。但在 Linux 中，无论操作系统管理几个磁盘分区，这样的目录树都只有一个。

Linux 使用标准的目录结构，在安装的时候，安装程序就已经为用户创建了文件系统和完整而固定的目录结构，并指定了每个目录的作用和其中的文件类型。

Root



2.2.2 目录功能简介

/bin 存放常用的二进制可执行命令，如：ls, mv, rm, mkdir, rmdir, gzip, tar, telnet, 及 ftp 等等。通常它与 /usr/bin 的内容是一样的。

/dev 存放与设备有关的特殊文件，基本上 Unix 或 Linux 系统都将设备当成文件，如 /dev/fd0 代表软盘，/dev/cdrom 则表示光盘。

/etc 存放系统管理和配置文件，如 LILO 的参数、用户的帐号和密码以及系统的主要设置。

/home 为用户设置的目录，比如用户 user 的主目录就是/home/user，可以用~user 表示。

/lib 标准程序设计库，又叫动态链接共享库，在 Linux 执行或编译内核时，均会用到。

/sbin 系统管理命令，这里存放的是系统管理员使用的管理程序，如： fdisk, mke2fs, fsck, mkswap, mount 等等。

/boot 放置 Linux 核心与启动和关闭系统有关的文档，一个在后面的实验中会使用的非常重要的目录。

/tmp 公用的临时文件存储点。

/root 系统管理员的主目录。

/mnt 系统提供这个目录是让用户临时装载其他的文件系统，如装载软盘的文件系统。

/lost+found 这个目录平时是空的，系统非正常关机时而留下的文件会放这里。类似于 windows 下的*.chk 文件。

/proc 虚拟的目录，是系统内存的映射。可直接访问这个目录来获取系统信息。

/var 这是系统在工作时，预先设置的工作目录，比方说各种服务的日志文件和收发的邮件等。

/usr 最庞大和最重要的目录之一，要用到的应用程序和文件几乎都在这个目录。其中包含：

/usr/X11R6 存放 X window 的目录。
/usr/bin 众多的应用程序。
/usr/sbin 超级用户的一些管理程序。
/usr/doc Linux 系统的说明文档（RedHat 7.0 以后改放在 /usr/share/doc 下）。
/usr/include Linux 下开发和编译应用程序所需要的头文件。
/usr/lib 存放常用的动态链接库和软件包的配置文件。
/usr/man 存放帮助文档（RedHat 7.0 以后放在 /usr/share/man 底下）。
/usr/src Linux 内核的源代码就放在这里，编译内核时必须用到。
/usr/local/bin 本地增加的命令，通常用于软件的升级。
/usr/local/lib 本地增加的库。

2.2.3 Linux 文件系统的装载和卸载

Linux 系统中每个分区都是一个文件系统，都有自己的目录层次结构。Linux 会将这些分属不同分区的、单独的文件系统按一定的方式形成一个系统的总的目录层次结构。

一．文件系统的装载

文件系统的装载是指将一个文件系统的顶层目录挂到另一个文件系统的子目录上，使它们成为一个整体，该子目录称为挂载点。装载前先要了解所要装载的文件系统的格式，看 Linux 是否支持该文件格式。装载时使用 mount 命令：

格式：mount [-参数] [设备名称] [装载点]

其中常用的参数有

-t<文件系统类型> 指定设备的文件系统类型，常见的有：

minix Linux 最早使用的文件系统
ext2 Linux 目前常用的文件系统
msdos MS-DOS 的 fat，就是 fat16
vfat windows98 常用的 fat32
nfs 网络文件系统
iso9660 光盘的标准文件系统
ntfs windows NT 2000 的文件系统
hpfs OS/2 文件系统
auto 自动检测文件系统

-o<选项> 指定装载文件系统时的选项，常用的选项如下：

codepage=XXX 代码页
iocharset=XXX 字符集
ro 以只读方式挂载
rw 以读写方式挂载
nouser 使一般用户无法挂载
user 可以让一般用户挂载设备

文件系统的装载点必须是一个已经创建好的目录。

装载软盘的命令：

```
# mk /mnt/floppy
# mount -t msdos /dev/fd0 /mnt/floppy
```

装载光盘的命令：

```
# mk /mnt/cdrom
```

mount -t iso9660 /dev/cdrom /mnt/cdrom

如果 windows98 装在 hda1 分区，要将它装载到 Linux 文件系统中：

mk /mnt/winc

mount -t vfat /dev/hda1 /mnt/winc

接下来就可以进入/mnt/floppy 、/mnt/cdrom 和/mnt/winc 目录访问这些文件系统了。

二．文件系统的卸载

umount 命令是 mount 命令的逆操作，umount 命令的作用是卸载一个文件系统。例如，将软盘装载到 /mnt/floppy 目录后，若要取出软盘，必须先使用 unmount 命令进行卸载，否则无法取下。它的参数使用方法和 mount 命令是一样的，命令格式如下：

格式：umount [-参数] [设备名称] [装载点]

2.3 Linux 系统的基本操作

2.3.1 Linux 系统的启动与退出

一．启动 Linux 系统

启动 Linux 系统很简单，只需直接加电就行了，但必须要输入用户的账号和口令。在系统安装过程中可以创建以下两种帐号：

1．root：超级用户帐号（供系统管理员使用），使用这个帐号可以在系统中做任何事情。

2．普通用户：这个帐号供普通用户使用，可以进行有限的操作。

一般的 Linux 使用者均为普通用户，而系统管理员则使用超级用户帐号完成一些系统管理的工作。如果只需完成一些由普通帐号就能完成的任务，建议不要使用超级用户帐号，以免无意中破坏系统，影响系统的正常运行。

用户登录分为两步：第一步，输入用户的登录名，系统根据该登录名识别用户；第二步，输入用户的口令，该口令是用户自己设置的一个字符串，对其他用户是保密的，是在登录时系统用来辨别真假用户的关键字。

当用户正确地输入用户名和口令后，就能合法地进入系统。屏幕会显示：

[root@loclhost /root] #

这时就可以对系统做各种操作了。注意超级用户的提示符是“#”，其他用户的提示符是“$”。

二． 退出系统

命令格式：shutdown［选项］时间［警告］

说明：Linux 是在需要关闭时必须告知的操作系统，不能只关掉电源。shutdown 将系统带到可以关闭电源的安全点。shutdown 命令可以安全地关闭或重启 Linux 系统，它在系统关闭之前给系统上的所有登录用户提示一条警告信息。该命令还允许用户指定一个时间参数，可以是一个精确的时间，也可以是从现在开始的一个时间段。精确时间的格式是 hh:mm，表示小时和分钟；时间段由“+”和分钟数表示。系统执行该命令后，会自动进行数据同步的工作。

时间：关闭系统的时间。关于完整的时间格式，请参考用户手册。

警告：向所有用户发出警告信息。

选项含义：

-t n：在向进程发出警告信号和杀掉信号之间等待 n 秒。

-k：不真正关闭系统，只向每人发送警告信息。

-r：关闭后重新启动。

-h 2：关闭后停机。

-n：快速关机，在重新启动和停机之前不作磁盘同步。

-f：快速重新启动，重新启动时不检查所有文件系统。

-c：取消已经运行的关闭命令。在本选项中，不能给出时间变量，但可以在命令行输入一个说明信息传给每个用户。

halt 命令也用来通知内核关闭系统，但它是一个只能由超级用户执行的命令。

如果只是退出登录，不论是 root 用户还是普通用户，只需简单地执行 exit 命令即可。

2.3.2 Linux 系统对文件和目录的操作命令

ls 显示目录内容

命令格式：ls [选项] [目录或是文件]

说明：对于每个目录，该命令将列出其中的所有子目录与文件。对于每个文件，ls 将输出其文件名及其所要求的其它信息。当未给出目录名或是文件名时，则显示当前目录的信息。该命令类似于 DOS 下的 dir 命令。

选项含义：

- a： 显示指定目录下所有子目录与文件，包括隐藏文件。

-d： 如果参数是目录，就只显示其名称而不显示其下的各文件。此选项往往与 l 选项一起使用，以得到目录的详细信息。

- i： 在输出的第一列显示文件的 i 节点号。

- R： 递归地显示指定目录的各个子目录中的文件。

- l： 以长格式来显示文件的详细信息。这个选项最常用。每行显示的信息依次为：文件类型与权限、链接数、文件属主、文件属组、文件大小、建立或最近修改的时间和文件名。

文件类型与权限为由 10 个字符构成的字符串表示，其中第一个字符表示文件类型，包括：-（普通文件）、d（目录）、l（符号链接）、b（块设备文件）和 c（字符设备文件）。后面的 9 个字符表示文件的访问权限，请参见后面的 chmod 命令。

对于符号链接文件，显示的文件名之后有"->"和引用文件路径名。 对于设备文件，其"文件大小"字段显示的是主、次设备号，而不是文件大小。

cp 文件或目录的复制

命令格式：cp [选项] 源文件或目录 目标文件或目录

说明：该命令是把指定的源文件复制到目标文件或把多个源文件复制到目标目录中。

选项含义：

- a： 该选项通常在拷贝目录时使用。它保留链接、文件属性，并且复制所有子目录。

- f： 删除已经存在的目标文件而不加提示。

- i： 和 f 选项相反，在覆盖目标文件之前会给出提示并要求用户确认。回答 y 时目标文件将被覆盖。为了用户文件的安全，最好使用该选项。

- r： 若给出的源文件是一目录文件，此时 cp 将递归复制该目录下所有的子目录和文件。此时目标文件必须为一个目录名。

mv　文件或目录更名或将文件由一个目录移到另一个目录中

命令格式：mv [选项] 源文件或目录 目标文件或目录

说明：根据 mv 命令中第二个参数类型的不同（是目标文件还是目标目录），mv 命令会将文件重命名或将其移至一个新的目录中。当第二个参数类型是文件时，mv 命令完成文件重命名。此时，源文件只能有一个（也可以是源目录名），它将所给的源文件或目录重命名为给定的目标文件名。当第二个参数是已存在的目录名称时，源文件或目录参数可以有多个，mv 命令将各参数指定的源文件均移至目标目录中。

选项含义：

- i： 询问方式操作。如果 mv 操作将导致对已存在的目标文件的覆盖，此时系统会询问是否重写，并要求用户回答 y 或 n，这样可以避免错误覆盖文件。

- f： 禁止询问操作。在 mv 操作要覆盖某个已有的目标文件时不给予任何提示，指定此选项后，i 选项将不再起作用。

rm　删除文件或目录

命令格式：rm [选项] 文件…

说明：该命令的功能为删除一个目录中的一个或多个文件或目录，它也可以将某个目录及其下的所有文件及子目录全部删除。

选项含义：

- f： 忽略不存在的文件，不给出提示。

- r： 指示 rm 将参数中列出的全部目录和子目录均递归地删除。如果没有使用- r 选项，则 rm 不会删除目录。

- i： 进行交互式删除。 使用 rm 命令要特别小心。因为一旦文件被删除，它是不能被恢复的。为了防止这种情况的发生，可以使用 i 选项来逐个确认要删除的文件。

mkdir　创建目录

命令格式：mkdir [选项] dir-name

说明：该命令创建由 dir-name 命名的目录。要求创建目录的用户在当前目录中（dir-name 的父目录中）具有写权限，并且 dirname 不能是当前目录中已有的目录或文件名。

选项含义：

- m： 对新建目录设置存取权限，也可以用 chmod 命令修改该权限。

- p： 可以是一个路径名称。此时若路径中的某些目录尚不存在，加上此选项后，系统将自动建立好那些尚不存在的目录，即一次可以建立多个目录。

rmdir　删除空目录

命令格式：rmdir [选项] dir-name

说明：dir-name 表示目录名。使用该命令可以从某个目录中删除一个或多个子目录项。需要特别注意的是，一个目录被删除之前必须是空的。rm - r dir 命令可代替 rmdir，但是有危险性。删除某目录时也必须具有对其父目录的写权限。

选项含义：

-p： 递归删除目录 dirname，当子目录被删除后，其父目录为空时，也一同被删除。如果整个路径被删除或者由于某种原因保留部分路径，则系统在标准输出上显示相应的信息。

cd　改变工作目录

命令格式：cd [directory]

说明：该命令将当前目录改变至 directory 所指定的目录。若没有指定 directory，则回到用户的主目录。为了改变到指定目录，用户必须拥有对指定目录的执行和读权限。

该命令可以使用通配符。

pwd　显示出当前工作目录的绝对路径

命令格式：pwd

cat　显示文件

命令格式：cat [选项] 文件列表

说明：如果没有指定文件或连字号（-），就从标准输入读取。

选项含义：

-b：计算所有非空输出行，开始为 1。

-n：计算所有输出行，开始为 1。

-s：将相连的多个空行用单一空行代替。

-V：显示除 LFD 和 TAB 以外的所有控制符，使用^作标志并在高位置的字符前放 M-。

-A：相当于-vET。

-E：在每行末尾显示$符号。

-T：用^I 显示 TAB 符号。

find　搜索文件

命令格式：find 目录列表 [选项]

选项含义：

-name 文件：告诉 find 要找什么文件；要找的文件包括在引号中，可以使用通配符（* 和？）

-perm 模式：匹配所有模式为指定数字型模式值的文件。不仅仅是读，写和执行，所有模式都必须匹配。如果在模式前是负号（-），表示采用除这个模式外的所有模式。

-type x：匹配所有类型为 x 的文件。x 是 c（字符特殊），b（块特殊），d（目录），p（有名管道），l（符号连接），s（套接文件）或 f（一般文件）。

-links n：匹配所有连接数为 n 的文件。

-size n：匹配所有大小为 n 块的文件（512 字节块，若 k 在 n 后，则为 1K 字节块）。

-user 用户号：匹配所有用户序列号是前面所指定的用户序列号的文件，可以是数字型的值或用户登录名。

find 命令支持多个条件进行组合（and、or、not）：其中用-a 表示 and（与），用-o 表示 or（或），用!表示 not（非）。如，在全盘查找一个名为 a.txt 的文件：$ find / -name a.txt；在 home 目录下查找属于用户 xf 的所有扩展名为 htm 的文件：$ find / -name *.htm -a -user xf。

grep 按指定模式查找文件

命令格式：grep [选项] 字符串 文件列表

选项含义：

-v：列出不匹配串的行。

-c：对匹配的行计数。

-l：只显示包含匹配的文件的文件名。

-n：每个匹配行只按照相对的行号显示。

-i：产生不区分大小写的匹配，缺省状态是区分大小写。

more　通用的按页显示

命令格式：more [选项] 文件名

选项含义：

-n：n 是整数，用于建立大小为 n 行长的窗口。窗口大小是在屏幕上显示多少行。

-c：用 more 给文本翻页时通过在最上面清除一行，然后再在最后写下一行的办法写入。通常，more 清除屏幕，再写每一行。

-d：显示 "Press space to cpntinue, 'q' quit"代替 more 的缺省提示符。

-f：计算逻辑行代替屏幕行。长行在屏幕上换行显示，通常被 more 计算为新的一行；-f 标志对长行的换行显示不计数。

-l：不处理^L（换页）字符。通常，more 处理^L 与窗口填满暂停一样。

-s：将多个空行压缩处理为一个。

-p：不滚屏，代替它的是清屏并显示文本。

-u：禁止加下划线。

文件名：希望用 more 显示的文件列表。

2.3.3　文档备份与压缩命令

tar　为文件和目录创建档案

命令格式：tar [主选项+辅选项] 文件或者目录。

说明：利用 tar，用户可以为某一特定文件创建档案（备份文件），也可以在档案中改变文件，或者向档案中加入新的文件。tar 最初被用来在磁带上创建档案。现在，用户可以在任何设备上创建档案，如软盘。利用 tar 命令，可以把一大堆的文件和目录全部打包成一个文件，这对于备份文件或将几个文件组合成为一个文件以便于网络传输是非常有用的。使用该命令时，主选项是必须要有的，由它确定 tar 的工作。

主选项含义：

c：创建新的档案文件。如果用户想备份一个目录或是一些文件，就要选择这个选项。

r：把要存档的文件追加到档案文件的末尾。例如用户已经作好备份文件，又发现还有一个目录或是一些文件忘记备份了，这时可以使用该选项，将忘记的目录或文件追加到备份文件中。

t：列出档案文件的内容，查看已经备份了哪些文件。

u：更新文件。就是说，用新增的文件取代原备份文件，如果在备份文件中找不到要更新的文件，则把它追加到备份文件的最后。

x：从档案文件中释放文件。

辅助选项含义：

b：该选项是为磁带机设定的。其后跟一数字，用来说明区块的大小，系统预设值为 20（20*512 bytes）。

f：这个选项通常是必选的，使用档案文件或设备。

k：保存已经存在的文件。例如我们把某个文件还原，在还原的过程中，遇到相同的文件，不会进行覆盖。

m：在还原文件时，把所有文件的修改时间设定为现在。

M：创建多卷的档案文件，以便在几个磁盘中存放。

v：详细报告 tar 处理的文件信息。如无此选项，tar 不报告文件信息。

z：用 gzip 来压缩/解压缩文件，加上该选项后可以将档案文件进行压缩，但还原时也一定要使用该选项进行解压缩。

gzip　压缩文件

命令格式：gzip [选项] 压缩（解压缩）的文件名

说明：gzip 是一个在 Linux 系统中经常使用的对文件进行压缩和解压缩的命令，既方便又好用。

各选项的含义：

-c：将输出写到标准输出上，并保留原有文件。

-d：将压缩文件解压。

-l：对每个压缩文件，显示下列字段：压缩文件的大小、未压缩文件的大小、压缩比和未压缩文件的名字。

-r：递归式地查找指定目录并压缩其中的所有文件或者是解压缩。

-t：测试，检查压缩文件是否完整。

-v：对每一个压缩和解压的文件，显示文件名和压缩比。

-num：用指定的数字 num 调整压缩的速度，-1 或 -fast 表示最快压缩方法（低压缩比），-9 或 -best 表示最慢压缩方法（高压缩比）。系统缺省值为 6。

uinzip　　展开 *.zip 文件

命令格式：unzip [选项] 压缩文件名.zip

说明：在 Linux 系统下可以用 unzip 展开在 Windows 下用压缩软件 winzip 压缩的文件，该命令用于解开扩展名为.zip 的压缩文件。

选项含义：

-x：用文件列表解开压缩文件，但不包括指定的 file 文件。

-v：查看压缩文件目录，但不解压。

-t：测试文件有无损坏，但不解压。

-d：目录，把压缩文件解到指定目录下。

-z：只显示压缩文件的注解。

-n：不覆盖已经存在的文件。

-o：覆盖已存在的文件且不要求用户确认。

-j：不重建文档的目录结构，把所有文件解压到同一目录下。

2.3.4　权限改变命令

chmod　改变文件或目录的访问权限

Linux 系统中的每个文件和目录都有访问许可权限，用它来确定谁可以通过何种方式对文件和目录进行访问和操作。文件或目录的访问权限分为只读（r），只写（w）和可执行（x）三种。有三种不同类型的用户可对文件或目录进行访问：文件所有者（u），同组用户（g）、其他用户（o）。所有者一般是文件的创建者，他可根据需要把访问权限设置为所需要的任何组合，确定另两种用户的访问权限。

每一文件或目录的访问权限都有三组，每组用三位表示，分别为文件属主的读、写和执行权限；与属主同组的用户的读、写和执行权限；系统中其他用户的读、写和执行权限。当用 ls -l 命令显示文件或目录的详细信息时，最左边的一列为文件的访问权限。例如：

$ ls -l sobsrc. tgz

-rw-r--r-- 1 root root 483997 Ju1 l5 17:3l sobsrc. tgz

横线代表空许可。r 代表只读，w 代表可写，x 代表可执行。第一个字符指定了文件类型。例如：

- rw- r-- r--

普通文件 文件主 组用户 其他用户

是文件 sobsrc.tgz 的访问权限，？表示 sobsrc.tgz 是一个普通文件；sobsrc.tgz 的属主有读写权限；与 sobsrc.tgz 属主同组的用户只有读权限；其他用户也只有读权限。

确定了一个文件的访问权限后，用户可以利用 Linux 系统提供的一组命令重新设置与权限和用户相关的操作。chmod 命令来重新设定不同的访问权限，用户用它控制文件或目录的访问权限。该命令有两种用法。一种是包含字母和操作符表达式的文字设定法；另一种是包含数字的数字设定法。

1. 文字设定法

命令格式：chmod [who] [+ | - | =] [mode] 文件名

选项含义：

操作对象 who 可以是下述字母中的任一个或者它们的组合：

u： 表示“用户（user）”，即文件或目录的所有者。

g： 表示“同组（group）用户”，即与文件属主有相同组 ID 的所有用户。

o： 表示“其他（others）用户”。

a： 表示“所有（all）用户”。它是系统默认值。

操作符号可以是：

+： 添加某个权限。

-： 取消某个权限。

=： 赋予给定权限并取消其他所有权限（如果有的话）。

设置 mode 所表示的权限可用下述字母的任意组合：

r： 可读。

w： 可写。

x： 可执行。

u： 与文件属主拥有一样的权限。

g： 与和文件属主同组的用户拥有一样的权限。

o： 与其他用户拥有一样的权限。

2. 数字设定法

命令格式：chmod [mode] 文件名

说明：我们必须先了解用数字表示的属性的含义：0 表示没有权限，1 表示可执行权限，2 表示可写权限，4 表示可读权限，然后将其相加。所以数字属性的格式应为 3 个从 0 到 7 的八进制数，其顺序是（u）（g）（o）。

例如，如果想让某个文件的属主有“读/写”二种权限，需要把 4（可读）+2（可写）＝6（读/写）。

chgrp 改变文件或目录所属组

命令格式：chgrp [选项] group filename

说明：该命令改变指定文件所属的用户组。其中 group 可以是用户组 ID，也可以是 /etc/group 文件中用户组的组名。文件名是以空格分开的要改变属组的文件列表，支持通配

符。如果用户不是该文件的属主或超级用户，则不能改变该文件的组。

选项含义：

-R： 递归地改变指定目录及其下的所有子目录和文件的属组。

chown 更改某个文件或目录的属主和属组

命令格式：chown [选项] 用户或组 文件

说明：chown 将指定文件的拥有者改为指定的用户或组。用户可以是用户名或用户 ID，组可以是组名或组 ID，文件是以空格分开的要改变权限的文件列表，支持通配符。

选项含义：

-R：改变指定目录及其下的所有子目录和文件的拥有者。

-v ：显示 chown 命令所做的工作。

2.3.5 Linux 与用户有关的命令

passwd 修改用户口令

命令格式： passwd [用户名]

出于系统安全考虑，Linux 系统中的每一个用户除了有其用户名外，还有其对应的用户口令。只有超级用户可以使用“passwd 用户名”修改其他用户的口令，普通用户只能用不带参数的 passwd 命令修改自己的口令。

su 改变用户权限

命令格式： su [使用者帐号]

说明：它可以让一个普通用户拥有超级用户或其他用户的权限，也可以让超级用户以普通用户的身份做一些事情。普通用户使用这个命令时必须有超级用户或其他用户的口令。如要离开当前用户的身份，可以输入 exit。若没有指定的使用者帐号，则系统预设值为超级用户 root。

2.3.6 Linux 系统管理命令

write 向用户发送信息

命令格式：write 用户帐号 [终端名称]

说明：write 命令的功能是向系统中某一个用户发送信息。

例如： $ write Guest hello

此时系统进入发送信息状态，用户可以输入要发送的信息，输入完毕，希望退出发送状态时，按组合键< Ctrl+c>即可。

mesg 指令

命令格式： mesg [y/n]

说明：mesg 命令用于设定是否允许其他用户用 write 命令给自己发送信息。如果允许别人给自己发送信息，后面跟 y；否则跟 n。对于超级用户，系统的默认值为 n；而对于一般用户系统的默认值为 y。 如果 mesg 后不带任何参数，则显示当前的状态是 y 还是 n.。

free 查看当前系统内存使用情况

命令格式： free [-b | -k | -m]

说明：free 命令的功能是查看当前系统内存的使用情况，显示系统中剩余的内存和已用的物理内存和交换内存，以及共享内存和被核心使用的缓冲区。

选项含义：

-b： 以字节为单位显示系统中剩余的内存和已用的物理内存和交换内存，以及共享内存和被核心使用的缓冲区。

-k： 以 K 字节为单位显示。

-m： 以兆字节为单位显示。

2.3.7 Linux 磁盘管理命令

df （Disk Filesystem） 检查文件系统的磁盘空间占用情况

命令格式：df [选项]

说明：检查文件系统的磁盘空间占用情况。可以利用该命令来获取硬盘被占用了多少空间，目前还剩下多少空间等信息。 df 命令可显示所有文件系统对 i 节点和磁盘块的使用情况。

选项含义：

-a： 显示所有文件系统的磁盘使用情况，包括 0 块（block）的文件系统，如/proc 文件系统。

-k： 以 k 字节为单位显示。

-i： 显示 i 节点信息，而不是磁盘块。用户可以知道每一个文件系统中有多少可用的 i-node、其中有多少已被使用、还剩余多少，以及它们所占的比例等整个硬盘的使用情况。

-t： 显示各指定类型的文件系统的磁盘空间使用情况。

-x： 列出不是某一指定类型文件系统的磁盘空间使用情况（与 t 选项相反）。

-T： 显示文件系统类型。

有时会出现这样的情况：某些硬盘的容量超过了 100％。这是因为 Linux 系统为超级用户保留了 10％的空间由其单独支配。也就是说，超级用户见到的硬盘容量将是 110％。这样的安排对于系统管理而言是有好处的，当硬盘被使用的容量接近 100%时系统管理员还可以正常工作。df 工具程序被广泛地用来生成文件系统的使用统计数据。它能显示系统中所有的文件系统的信息，包括它们的总容量、可用的空闲空间、目前的安装点等。

du 显示目录（或文件）所占磁盘空间的大小

命令格式：du [选项] [Names…]

说明：统计目录（或文件）所占磁盘空间的大小。

[Names…]：目录名或文件名。该命令逐级进入指定目录的每一个子目录并显示该目录占用文件系统数据块（1024 字节）的情况。若没有给出 Names，则对当前目录进行统计。

选项含义：

-s： 对每个 Names 参数只给出占用的数据块总数。

-a： 递归地显示指定目录中各文件及子孙目录中各文件占用的数据块数。若既不指定-s，也不指定-a，则只显示 Names 中的每一个目录及其中的各子目录所占的磁盘块数。

-b： 以字节为单位列出磁盘空间使用情况（系统缺省以 k 字节为单位）。

-k： 以 k 字节为单位列出磁盘空间使用情况。

-c： 最后再加上一个总计（系统缺省设置）。

-l： 计算所有的文件大小，对硬链接文件，则计算多次。

-x： 跳过在不同文件系统上的目录不予统计。

dd　将输入文件复制到指定的输出文件中

命令格式：dd [选项]

说明：把指定的输入文件复制到指定的输出文件中，并且在拷贝过程中可以进行格式转换。可以用该命令实现 DOS 下的 diskcopy 命令的作用。先用 dd 命令把软盘上的数据写成硬盘的一个临时文件，再把这个临时文件写入第二张软盘上，完成 diskcopy 的功能。需要注意的是，应该将硬盘上的临时文件用 rm 命令删除掉。系统默认使用标准输入文件和标准输出文件。

选项含义：

if = 输入文件（或设备名称）。

of = 输出文件（或设备名称）。

ibs = bytes 一次读取 bytes 字节，即读入缓冲区的字节数。

skip = blocks 跳过读入缓冲区开头的 ibs*blocks 块。

obs = bytes 一次写入 bytes 字节，即写入缓冲区的字节数。

bs = bytes 同时设置读/写缓冲区的字节数（等于设置 ibs 和 obs）。

cbs = byte 一次转换 bytes 字节。

count=blocks 只拷贝输入的 blocks 块。

conv = ASCII 把 EBCDIC 码转换为 ASCII 码。

conv = ebcdic 把 ASCII 码转换为 EBCDIC 码。

conv = ibm 把 ASCII 码转换为 alternate EBCDIC 码。

conv = ucase 把字母由小写转换为大写。

conv = lcase 把字母由大写转换为小写。

conv = swab 交换每一对输入字节。

conv = noerror 出错时不停止处理。

conv = sync 把每个输入记录的大小都调到 ibs 的大小（用 NULL 填充）。

fdformat　格式化软盘

命令格式：format [-n] device

说明：-n 软盘格式化后不作检验。

device 指定要进行格式化的设备，通常是下述设备之一：

/dev/fd0h1200

/dev/fd0D720

/dev/fd0H720

/dev/fd0H1440

2.3.8　进程管理命令

at　在指定时间执行程序

命令格式：　at　time [day] [file]

说明：用户使用 at 命令在指定时刻执行指定的命令序列。也就是说，该命令至少需要指定一个命令、一个执行时间才可以正常运行。at 命令可以只指定时间，也可以时间和日期一起指定。在任何情况下，超级用户都可以使用这个命令。对于其他用户来说，是否可以使用就取决于两个文件： /etc/at.allow 和 /etc/at.deny

例：at　10:30 5/26/2004　who

bg 和 fg　进程的挂起及恢复命令

命令格式：bg/fg

说明：bg 命令用来迫使被挂起的进程在后台运行。例如，当你已经在前台启动了一个命令时（没有在此命令后使用&），你才想到这一命令将运行较长一段时间,但你这时还需使用 shell。在这种情况下，可通过 ctrl+z 挂起当前运行的进程。此时你既可以使它长期挂起，也可以通过输入 bg 把这一进程放到后台运行。这样 shell 就可以用来执行其他的命令了。

fg 命令用来激活某个被挂起的进程并使它在前台运行。当有一个进程正在运行时，由于某种原因需要挂起它，在执行完其他任务后，需要重新把这一进程调到前台运行，这时便可用 fg 命令使这一进程继续运行。

who 查看当前在线上的用户情况

命令格式：who

说明：该命令主要用于查看当前在线上的用户情况，系统管理员可以通过该命令监视每个登录用户的情况。

w　显示目前登录的用户及正在执行的命令

命令格式：w

说明：w 命令与 who 命令相比功能更加强大。w 命令的显示项目按以下顺序排列：当前时间，系统启动到现在的时间，登录用户的数目，系统在最近 1 秒、5 秒和 15 秒的平均负载。最后是每个用户的各项数据，各项数据显示顺序如下：登录帐号、终端名称、远程主机名、登录时间、空闲时间、JCPU（和该终端连接的所有进程占用的时间）、PCPU（当前进程所占用的时间）、当前正在运行进程的命令行。

ps　进程查看命令

命令格式：　ps [选项]

说明：该命令可以确定有哪些进程正在运行以及运行的状态、进程是否结束、哪些进程占用了过多的资源等等。ps 命令最常用的还是用于监控后台进程的工作情况。

选项含义：

-e　显示所有进程

-f　全格式

-h　不显示标题

-l　长格式

-w　宽输出

-a　显示终端上的所有进程，包括其他用户的进程

-r　只显示正在运行的进程

-x　显示没有控制终端的进程

若按长格式输出，则显示以下内容：

PID：进程号

PRI：进程优先级。

NI：Linux 进程的 nice 值。负数意味着占用较少的 CPU 时间。

SIZE：虚拟映象的大小，大小的计算为文本+数据+栈。

RSS：驻留空间的大小。显示当前常驻内存的程序的 K 字节数。

WCHAN：进程等待的内核事件名。

STAT：进程状态，用这些代码中的一个给出：R（可执行的）、S（睡眠状态）、D（不间断睡眠）、T（停止或跟踪）、Z（僵尸）。

W：进程没有驻留页。

TT：进程的控制 tty 名。

PAGEIN：造成从磁盘读取页的页面出错负。

TRS：文本驻留大小。

SWAP：交换设备上的 K 字节数。

kill　向指定的进程发送信号

命令格式：　kill [-signal]　pid

说明：kill 命令可以终止后台进程。kill 命令是通过向进程发送指定的信号来结束进程的。如果没有发送指定信号，那么默认值为 SIGTERM 信号。SIGTERM 信号将终止所有不能捕获该信号的进程。至于那些可以捕获该信号的进程可能就需要使用 kill（9）信号了，该信号是不能被捕捉的。

选项含义：

-signal　发送的信号类型，默认值为 15，即 SIGTERM 信号。

2.3.9　Linux 其它命令

echo　显示字符串

命令格式：　echo [-n] 字符串

说明：选项 n 表示输出文字后不换行；字符串可以加引号，也可以不加引号。用 echo 命令输出加引号的字符串时，将字符串原样输出；用 echo 命令输出不加引号的字符串时，将字符串中的各个单词作为字符串输出，各字符串之间用一个空格分割。

cal　显示日历

命令格式：　cal [选项]　[月份]　[年]

选项含义：

- j：　显示出给定月份中的每一天是一年中的第几天（从 1 月 1 日算起）。

- y：　显示出整年的日历。

date　显示和设置系统日期和时间

命令格式：　date [选项] 显示时间格式（以+开头，后面接格式）

选项含义：

-d datestr：　显示由 datestr 描述的日期。

-s datestr：　设置 datestr 描述的日期。

-u：　显示或设置通用时间。

时间域

% H：　小时（00..23）

% I：　小时（01..12）

% k：　小时（0..23）

% l：　小时（1..12）

% M：　分（00..59）

%p： 显示出 AM 或 PM
%r： 时间（hh：mm：ss AM 或 PM），12 小时
%s： 从 1970 年 1 月 1 日 00：00：00 到目前经历的秒数
%S： 秒（00..59）
%T： 时间（24 小时制）（hh:mm:ss）
%X： 显示时间的格式（%H:%M:%S）
%Z： 时区 日期域
%a： 星期几的简称（Sun..Sat）
%A： 星期几的全称（Sunday..Saturday）
%b： 月的简称（Jan..Dec）
%B： 月的全称（January..December）
%c： 日期和时间（Mon Nov 8 14：12：46 CST 1999）
%D： 日期（mm／dd／yy）
%m： 月（01..12）
%x： 显示日期的格式（mm/dd/yy）
%y： 年的最后两个数字（1999 则是 99）
%Y： 年（例如：1970，1996 等）

clear 清屏
命令格式： clear

2.4 vi 的使用

vi 是 Unix 和 Linux 操作系统使用的全屏幕文本编辑器，任何一台安装了 Unix 或 Linux 的机器都会提供这套软件，它是系统管理员手中得力的工具。vi 的用法和 DOS 下的文本编辑器有较大的区别，刚开始使用时可能不太习惯，但这是我们的实验系统配置的编辑器，应该要学会掌握。

2.4.1 vi 的操作模式

vi 有三种操作状态：命令模式（Command mode）、插入模式（Insert mode）和末行命令模式（Last line mode）。它们的功能如下：

1． 命令模式：当执行 vi 后，首先会进入指令模式，此时输入的任何字符都被视为指令。命令模式用于控制屏幕光标的移动，文本的删除，移动复制某区段，进入插入模式下，或者进入末行命令模式。

2． 插入模式：在命令模式下输入相应的插入命令进入该模式。只有在插入模式下，才可做文字数据输入，按 Esc 可回到命令模式。

3． 末行命令模式：在命令模式下输入某些特殊字符，如/、？和：，可进入末行命令模式。在该模式下可储存文件或离开编辑器，也可设置编辑环境，如寻找字符串、列出行号等。

2.4.2 vi 的进入与退出

一．进入 vi
若要编辑文件 myfile，执行如下指令即可：

%vi　myfile

屏幕显示 vi 的编辑窗口，进入命令操作模式。

也可以直接键入 vi ，在退出时保存文件。

二．退出 vi

如果在输入模式下，则先利用 ESC 进入指令模式，而后即可选用下列指令退出 vi：

：q!　　离开 vi，并放弃刚才编辑的内容。

：wq　　存盘并退出

：ZZ　　存盘并退出

：x　　同 wq

：w　　存盘但并不退出

：q　　退出 vi，若文件被修改过，则会被要求确认是否放弃修改的内容。此指令可与 w 配合使用。

注意：如果不知道现在是处于什么模式，则可以多按几次 ESC 键，以便确定进入指令模式。

2.4.3　vi 的常用命令

一．命令模式的常用命令

在命令模式下可以进行文本的编辑工作。使用下列命令，配合一般键盘上的功能键，如方向键、[Insert] 、[Delete]等等，就可以利用 vi 来处理文字资料了。当然 vi 还提供其他许许多多的功能让文字的处理更加方便、快捷。

1．　光标的移动

命令	说明	功能键
h	向左移一个字符	[←]
l	向右移一个字符	[→]
j	向上移一个字符	[↑]
k	向下移一个字符	[↓]
0	移至该行之首	[Home]
$	移至该行之末	[End]
^	移至该行的第一个非空白字符处	
H	移至窗口的第一行	
M	移至窗口的中间那行	
L	移至窗口的最后一行	
G	移至该文件的最后一行	
nG	移至该文件的第 n 行	
[Ctrl+f]	向后翻一页	[PageDown]
[Ctrl+b]	向前翻一页	[PageUp]

2．删除与修改

命令	说明	功能键
x	删除光标后的字符	[Delete]
X	删除光标前的字符	
dd	删除光标所在的行	

ndd	删除包括光标所在行的 n 行文本
r	修改光标所在字符
R	进入替换状态，直到按 ESC 回到指令模式为止。[Insert]
s	删除光标所在字符，并进入输入模式
S	删除光标所在的行，并进入输入模式
u	恢复刚才被修改的文本
U	恢复光标所在行的所有修改
.	重复上一次命令的操作

3． 复制

命令	说明
Y	复制当前行至编辑缓冲区
nY	复制当前行开始的 n 行至编辑缓冲区
p	将编辑缓冲区的内容粘贴到光标后的一行
P	将编辑缓冲区的内容粘贴到光标前的一行

二．插入模式的常用命令

命令	说明
a	从光标所在位置后面开始新增文本
A	从光标所在行最后面的地方开始新增文本
i	从光标所在位置前面开始插入文本
I	从光标所在列的第一个非空白字元前面开始插入文本
o	在光标所在列下新增一行并进入输入模式
O	在光标所在列上方新增一行并进入输入模式

三．末行命令模式的常用命令

命令	说明
:q	结束编辑
:q!	强制离开 vi，放弃存盘
:w	存盘
:w filename	将编辑内容存为名为 filename 的文件
:wq	存盘并退出
ZZ	存盘并退出（这属于命令模式）
:x	若有修改存盘，退出程序
:e filename	编辑名为 filename 的文件
:set nu	显示行号
:set nonu	不显示行号
/exp	往前查找字符串 exp
?exp	往后查找字符串 exp

vi 是一个功能强大的编辑器，以上只罗列了一些常用命令，若想了解更多的操作方法，请参阅相关的参考书籍。

2.5 Linux 的编译器 gcc

C 语言是 Linux 下的最常用的程序设计语言，Linux 上的很多应用程序就是用 C 语言写的。我们实验中的编程都是使用 C 或 C++来实现的。Linux 系统上运行的 GNU C 编译器（GCC）是一个全功能的 ANSI C 兼容编译器。虽然 GCC 没有集成的开发环境，但堪称是目前效率很高的 C/C++编译器。

GCC 安装后目录结构：

/usr/lib/gcc-lib/target/version/ （及子目录）编译器就在这个目录下。

/usr/bin/gcc 可以从命令行执行的二进制程序在这个目录下。

/usr/target/(bin|lib|include)/ 库和头文件在这个目录下。

/lib/、/usr/lib 和其他目录，系统的库在这些目录下。

命令格式： gcc [选项] 源文件 [目标文件]

选项含义：

-o FILE：指定输出文件名，在编译为目标代码时，这一选项不是必须的。如果 FILE 没有指定，缺省文件名是 a.out.

-c：GCC 仅把源代码编译为目标代码。缺省时 GCC 建立的目标代码文件有一个 .o 的扩展名。

-static：链接静态库，即执行静态链接

-O：GCC 对源代码进行基本优化。这些优化在大多数情况下都会使程序执行得更快。

-ON：指定代码优化的级别为 N,o<=N<=3。-O2 选项告诉 GCC 产生尽可能小和尽可能快的代码。-O2 选项将使编译的速度比使用 -O 时慢， 但通常产生的代码执行速度会更快。

-g：在可执行程序中包含标准调试信息。GCC 产生能被 GNU 调试器使用的调试信息，以便调试你的程序。GCC 提供了一个很多其他 C 编译器里没有的特性, 在 GCC 里可以使 -g 和 -O （产生优化代码）联用。

-pedantic：允许发出 ANSI/ISO C 标准所列出的所有警告

-pedantic –errors：允许发出 ANSI/ISO C 标准所列出的所有错误

-w ：关闭所有警告，建议不要使用此项

-Wall：允许发出 gcc 能提供的所有有用的警告，也可以用-W（warning）来标记指定的警告

-MM：输出一个 make 兼容的相关列表

-v：显示在编译过程中的每一步用到的命令

2.6 shell 程序设计

shell 是用户和 Linux 操作系统之间的接口，它是命令语言、命令解释程序及程序设计语言的统称。用户在提示符下输入的命令都由 shell 先加以解释然后再传给 Linux 核心。shell 的另一个重要特性是它自身就是一个解释型的程序设计语言，shell 程序设计语言支持绝大多数在高级语言中能见到的程序元素，如函数、变量、数组和程序控制结构。Linux 中有多种 shell，如 ash、bash、ksh、csh、zsh 等，其中缺省使用的是 bash。

命令解释程序通常处于操作系统的最外层，用户一般直接与之打交道。它的主要功能是解释用户输入的命令，并转入相应的命令处理程序去执行。各种不同操作系统的命令解释程序的工作流程不同。以 UNIX 为例，shell 作为操作系统的最外层直接与用户交互。当用户

打开终端后，系统为该终端建立一个称为 shell 的进程，由该进程去读入、识别和执行用户键入的各种命令。

当普通用户成功登录，系统将执行一个称为 shell 的程序。正是 shell 进程供了命令行提示符。作为默认值，对普通用户以“$”作提示符，对超级用户（root）以“#”作提示符。使用 shell 编程类似于 DOS 中的批处理文件，称为 shell script，又叫 shell 程序或 shell 命令文件。

2.6.1 shell 程序的编写和执行

一、编写 shell 程序

用户可以用任何编辑程序来编写 shell 程序，因为 shell 程序是解释执行的，所以不需要编译装配成目标程序。在我们的实验系统中，大家可使用 vi 来完成这项工作。例如编写如下的 shell 程序：

```
$vi  first
#！/bin/bash
# My first shell script
#
clear
echo "Hello, everybody!"
```

程序 first 的功能是清屏并显示字符串“Hello, everybody!”。程序的第一行一般为“#！/bin/bash”，由#开始的行为注释行，叹号“！”告诉 shell 运行叹号之后的命令并用文件的其余部分作为输入，也就是运行/bin/bash 并让/bin/bash 去执行 shell 程序的内容。

三. 执行 shell 程序

执行 shell 程序的方法有如下三种：

1．将 shell 程序作为 sh 的输入

命令格式：bash shell 程序文件名

实际上这是调用一个新的 bash 命令解释程序，shell 程序文件名只是作为参数传递给它。新启动的 shell 将去读指定的文件，执行文件中列出的命令，当所有的命令都执行完成后结束。该方法的优点是能够利用 shell 的调试功能。

例：$sh first

2．利用输入重定向

命令格式：bash<SHELL 程序文件名

这种方式就是利用输入重定向，使 shell 命令解释程序的输入取自指定的程序文件。

3．直接执行

由于 shell 程序是用 vi 编写的文本文件，系统赋予的许可权限都是 644(rw-r-r--)，需要用 chmod 命令使 shell 程序成为可执行的，然后只需要直接键入文件名即可。

2.6.2 shell 基础

一. I/O 重定向

标准输入（stdin，文件描述指针为 0）、标准输出（stdout，文件描述指针为 1）和标准错误输出（stderr，文件描述指针为 2）是 Linux 的三个特殊文件，由系统自动打开。一般情况下，shell 通过标准输入接受用户的命令，将执行的结果送往标准终端显示输出，而将出错信息写到标准错误输出。

如果用户希望从文件中接收输入的命令和参数，或者把输出结果写入某个文件，则可以利用 Linux 的输入/输出重定向“>”、“>>”“<”和“<<”来完成。

1．输出重定向

命令格式：command>file

功能：将命令的输出结果重定向到名为 file 的文件中。

例：ls –l > fileinfo

将当前目录下的文件信息以长格式的方式保存在文件 fileinfo 中。文件原来的内容会被覆盖。

命令格式：command>>file

功能：将标准输出的结果追加到名为 file 的文件后。

例：date >> fileinfo

将日期信息以追加的方式保存在文件 fileinfo 后面。

2．输入重定向

命令格式：command<file

功能：从文件 file 中接收命令所需的信息。

例：wc<< fileinfo

统计文件 fileinf 中的行数、单词数和字符数。

命令格式：command<<string

功能：当输入的行为 string 时，结束命令的输入。

例：wc<<hello

统计输入“hello”一行前的行数、单词数和字符数。

3．错误输出重定向

命令格式：command>&file

功能：将命令的标准错误输出全部重定向到名为 file 的文件中。

命令格式：command>>&file

功能：将标准输出和标准错误输出的结构都追加到文件 file 后。

二．管道 pipe

当后一命令的输入是前一命令的输出时，利用管道 pipe 即可方便地完成信息传递工作。管道起的是数据传送的作用，它是输出重定向和输入重定向的组合，用它可以简化命令。

命令格式：command1| command2[| command3...]

功能：将命令 1 的输出作为命令 2 的输入（如果后面还接有命令，则再将命令 2 的输出作为命令 3 的输入）。

例：who | sort > user_list

将 who 命令的结果排序后，写入文件 user_list 中。

三．后台进程

在 Linux 中可以把执行时间很长的进程转入后台，不占据终端。

命令格式：command &

功能：产生一个后台的进程，此进程在后台运行的同时，可以输入其他的命令。命令输入后，系统显示该进程的标识（pid），在进程运行期间，可用 ps 命令查看到该进程，也可用 kill-9 命令终止其执行。若命令执行完毕，系统将显示结束信息。

四．通配符

Linux 下常用的通配符如下：

*　匹配任意字符串

？　匹配任意一个字符

[...] 匹配方括号内的字符组。

例 1：ls *.c

显示所有扩展名为 c 的文件。

例 2：ls a*

显示所有以字母 a 打头的文件。

例 3：ls file?

显示所有前四个字符为 file，文件名长度为 5 的文件，如 file1、file2、filea 等。

例 4：ls [abc]

显示所有第一个字符为 a、b 或 c 的文件。

例 5：ls /bin/[a-d]*

显示/bin 目录下所有第一个字符为 a 到 d 的文件。

五．shell 程序的变量和参数

shell 提供说明和使用变量的功能。Shell 程序设计语言的变量类型很简单，只有字符类型的变量，即所有变量的取值都是一个字符串。

1．Shell 变量定义的规则

变量是以字母或下划线开始的字符串，它是大小写敏感的。shell 程序采用$var 的形式来引用名为 var 的变量的值。定义的格式为：

变量名 = 变量值

例：$no=10

$hello=“Hello everybody!”

$echo $no

$echo $hello

程序执行的结果是在屏幕上显示“10”和“Hello everybody!”

注意：在进行变量赋值时，等号两边不能有空格。若变量中本身就包含了空格，则整个字符串都要用双引号括起来。

2．环境变量

除了用户在编程时定义的变量外，shell 还定义了一些和系统的工作环境有关的变量，对这些变量用户还可以重新定义，常用的 shell 环境变量如下：

HOME：用于保存注册目录的完全路径名。

PATH：用于保存用冒号分隔的目录路径名，shell 将按 PATH 变量中给出的顺序搜索这些目录，找到的第一个与命令名称一致的可执行文件将被执行。

TERM：终端的类型。

UID：当前用户的标识符，取值是由数字构成的字符串。

PWD：当前工作目录的绝对路径名，该变量的取值随 cd 命令的使用而变化。

PS1：主提示符，在特权用户下，缺省的主提示符是“#”，在普通用户下，缺省的主提示符是“$”。

例：如果要显示 home 目录的位置，使用如下命令：

```
$ echo $HOME
```

3．位置变量

位置变量类似于 C 语言的命令行参数，是指在 shell 程序名之后的参数，用$n 表示。位置变量之间用空格分隔，第一个位置变量名为$1，第二个为$2，依次类推。$0 是一个特殊的变量，它是当前 shell 程序的文件名。位置参数的个数用$#表示。

4．预定义变量

预定义变量是指在 shell 一开始时就定义了的变量，用户根据需要来使用这些变量，但不能重定义它们。所有预定义变量都是由$符和另一个符号组成的，常用的 shell 预定义变量有：

$?	前一命令执行后返回的状态
$$	当前进程的 pid
$!	最近运行的后台进程的 pid
$0	当前执行的进程名

2.6.3 shell 程序设计

除了前面介绍的各项功能外，shell 也和其他高级程序设计语言一样，能够提供用来控制程序执行流程的命令，用户可以用这些命令建立非常复杂的程序。

一．test 命令

test 是最常用的一个测试命令，用于测试后面的表达式的值是否为“真”，若其值为“真”，返回 0，否则返回非 0 值。shell 用于指定条件值的不是布尔表达式而是命令和字符串。test 命令可以进行数值、字符和文件三个方面的测试。

1．文件测试

-e filename	如果文件存在
-s filename	如果文件存在且非 0
-r filename	如果文件存在且可读
-w filename	如果文件存在且可写
-x filename	如果文件存在且可执行
-d filename	如果文件存在且为目录
-f filename	如果文件存在且为普通文件
-c filename	如果文件存在且为字符型设备文件
-b filename	如果文件存在且为块设备文件

例：检测从命令行输入的文件是否存在。

```
#!/bin/bash
```

```
if [ $# -ne 1 ]
then
    echo "Usage - $0    file-name"
    exit 1
fi
if [ -f $1 ]
then
    echo "$1 file exist"
else
    echo "Sorry, $1 file does not exist"
fi
```

2．数值测试

test n1 –eq n2	整数 n1 和 n2 相等
test n1 –ne n2	整数 n1 和 n2 不相等
test n1 –gt n2	整数 n1 大于 n2
test n1 –ge n2	整数 n1 大于或等于 n2
test n1 –lt n2	整数 n1 小于 n2
test n1 –le n2	整数 n1 小于或等于 n2

3．字符串测试：

test s	字符串 s 非空
test s1=s2	字符串 s1 等于 s2
test s1!=s2	字符串 s1 不等于 s2
test –z s	字符串 s 为空串

另外，Linux 还提供了与（!）、或（-o）、非（-a）三个逻辑操作符用于将测试条件连接起来，其优先级为：“!”最高，“-a”次之，“-o”最低。

二．表达式求值

shell 变量都是字符类型，但如果是数字字符，也能进行简单的算术运算，格式如下：

```
$ expr 1 + 3
$ expr 2 – 1
$ expr 10 / 2
$ expr 20 % 3
$ expr 10 \* 3
$ echo `expr 6 + 3`
```

在上述表达式中，数字和运算符之间要有空格；使用“*”号时要加上转义字符“\”；最后一条命令中，“expr 6 + 3”前后使用的是反引号“`”。

三．条件控制

1．if 语句

if 语句的格式：

if 条件

```
    then      命令行
   else      命令行
   fi
```

例：显示程序的位置变量

```
#!/bin/sh
if [$# = 2]
  then echo $1 $2
  else echo $0 usage : Enter 2 arguments!
```

2． case 语句

利用 case 语句可根据字符串或变量的值从多个选项中选择一项来执行，其格式如下：

```
case string in
exp1)      命令行 1;;
exp2)      命令行 2;;
……
*)         其他命令行;;
esac
```

shell 通过计算字符串 string 的值，将其结果依次和表达式 exp1、exp2 等进行比较，直到找到一个匹配的表达式为止，如果找到了匹配项则执行它下面的命令行。当找不到任何相应的匹配项时，shell 就会执行“其他命令行”的命令。在 case 表达式中也可以使用 shell 的通配符（“*”、“？”、“[ ]”）。

例：从命令行输入一个简单的算术表达式，计算结果。

```
#!/bin/sh
if test $# = 3
then
 case $2 in
  +) let z=$1+$3;;
  -) let z=$1-$3;;
  /) let z=$1/$3;;
  x|X) let z=$1*$3;;
  *) echo Warning - $2 invalied operator, only +,-,x,/ operator allowed
      exit;;
 esac
 echo Answer is $z
else
 echo "Usage - $0      value1 operator value2"
fi
```

四．循环

1．for 循环

for 循环对一个变量的可能的值都执行一个命令序列。for 循环的一般格式为：

```
for   变量名   [in 数值列表]
do
```

命令行
done

变量名可以是用户选择的任何字符串，如果变量名是 var，则在 in 之后给出的数值将顺序替换循环命令列表中的$var。如果省略了 in，则变量 var 的取值将是位置参数。

例：循环 5 次显示字符串

```
#!/bin/sh
for i in 1 2 3 4 5
do
echo "Welcome $i times"
done
```

2．while 循环

while 循环是根据命令执行的返回状态值来控制循环的，若“命令行 1”中最后一个命令的返回状态为真，执行 do...done 之间的“命令行 2”。一般格式为：

```
while       命令行 1
      do
      命令行 2
done
```

例：下列程序显示 number = 10;number=20;……;一直到 number=100;

```
#!/bin/sh
i=1
while [ $i -le 10 ]
do
  echo “number = `expr $i \* $n`; ”
i=`expr $i + 1`
done
```

3．until 循环

until 命令是和 while 相似的循环，但 until 循环是“命令行 1”中最后一个命令的返回状态为假时继续执行循环。

其格式如下：

```
until   命令行 1
do
      命令行 2
done
```

4．break 和 continue 语句

break 和 continue 语句的用法和在 C 语言中的类似，break 用于立即终止当前循环的执行，而 continue 用于不执行循环中后面的语句而立即开始下一个循环的执行。这两个语句只有放在 do 和 done 之间才有效。

五．函数定义

在 shell 中还可以定义函数。函数实际上也是由若干条 shell 命令组成的，因此它与 shell 程序形式上是相似的，不同的是它不是一个单独的进程，而是 shell 程序的一部分。函数定

义的基本格式为：

```
functionname()
{
  命令行
}
```

调用函数的格式为：

```
functionname param1 param2……
```

例：编写的显示今天日期的函数如下：

```
today()
{
echo This is a `date +"%A %d in %B of %Y (%r)"`
return
}
```

shell 函数可以完成某些例行的工作，而且还可以有自己的退出状态，因此函数也可以作为 if、while 等控制结构的条件。

在函数定义时不用带参数说明，但在调用函数时可以带有参数，此时 shell 将把这些参数分别赋予相应的位置参数$1、$2、...及$*。

2.6.4 shell 程序的调试

shell 是一种解释执行的语言，可以利用 bash 命令解释程序的选择项来调试程序。调用 bash 的形式是：

bash -选择项 shell 程序文件名

选择含义如下：

-e：如果一个命令失败就立即退出

-n：读入命令但是不执行它们

-u：置换时把未设置的变量看作出错

-v：当读入 shell 输入行时把它们显示出来

-x：执行命令时把命令和它们的参数显示出来

上面的所有选项也可以在 shell 程序内部以“set -选择项”的形式引用，而“set +选择项”则将禁止该选择项起作用。如果只想对程序的某一部分使用某些选择项时，则可以将该部分用上面两个语句包围起来。

1．未置变量退出和立即退出

未置变量退出特性允许用户对所有变量进行检查，如果引用了一个未赋值的变量就终止 shell 程序的执行。shell 通常允许未设置变量的使用，在这种情况下，变量的值为空。如果设置了未置变量退出选择项，则一旦使用了未置变量就显示错误信息，并终止程序的运行。未置变量退出选择项为“-u”。

当 shell 运行时，若遇到不存在或不可执行的命令、重定向失败或是命令非正常结束等情况时，如果未经重新定向，该出错信息会打印在终端屏幕上，而 shell 程序仍将继续执行。要想在错误发生时迫使 shell 程序立即结束，可以用“-e”选项将 shell 程序的执行立即终止。

2．shell 程序的跟踪

调试 shell 程序的主要方法是利用 shell 命令解释程序的“-v”或“-x”选项来跟踪程序

的执行。“-v”选择项使程序在执行过程中，将它读入的每一个命令行都显示出来。而“-x”选择项使 shell 在执行程序的过程中，把它执行的每一个命令在行首用一个“+”加上命令名显示出来，并且把每一个变量和该变量的值也显示出来。因此，它们的主要区别在于：在执行命令行之前无“-v”则打印出命令行的原始内容，而有“-v”则打印出经过替换后的命令行的内容。

除了使用 shell 的“-v”和“-x”选择项以外，还可以在 shell 程序内部采取一些辅助调试的措施。例如，可以在 shell 程序的一些关键地方使用 echo 命令把必要的信息显示出来，它的作用相当于 C 语言中的 printf 语句，这样就可以知道程序运行到什么地方及程序目前的状态。

3．在编辑过程中执行程序

在使用 vi 编辑 shell 程序时，保存修改的结果后，便可在末行命令方式下执行该程序，其命令如下：

：！filename