与UNIX有关的几个名词

多用户(Multi-users)允许多个用户同时使用计算机的能力。

多任务(Multi-tasking)为每一个用户同时执行多个任务的能力。

终端（Terminal）终端是用户和计算机进行实时交互的接口设备。终端一般由显示器、键盘和终端控制器组成，有的还带有鼠标。显示器和键盘与终端控制器相连，终端控制器通过通用端口（如串口）和计算机主机相连。用户通过终端和计算机交流。一台UNIX主机可以连多个终端。终端分为两类：字符终端和图形终端。

用户名(User)　UNIX是多用户操作系统，它允许多个用户同时使用。每个用户拥有用户名、登录口令以及操作特权。用户每次使用UNIX时必须先登录：输入用户名和口令。一般用户的UNIX操作系统提示符一般为"$"。

工作组(Group)　几个用户可以组成一组，同组内的用户可以共享信息。

root用户　UNIX的超级用户，拥有其他用户没有的特权。root用户的UNIX操作系统提示符一般为"#"。

进程(process) 是正在执行的程序。UNIX允许多个进程同时存在，每个进程都有唯一代号称为进程标识符(pid --- process id)。

前台进程(foreground process) 可以和用户直接进行人机交互的进程。前台进程可以接收键盘输入并将结果显示在显示器上。缺省状态下用户启动的UNIX程序运行在前台。

后台进程(background process) 不直接和用户进行交互的进程。用户一般是感觉不到后台进程的运行。UNIX有很多系统进程在后台执行。

设备(device) 或称设备文件，代表某个硬件，如磁盘、软盘、串口。UNIX对硬件的存取操作都是通过设备文件来完成的。设备文件放在/dev目录下，如/dev/hd0表示第一个软盘驱动器。

Linux命令行接口

启动终端

启动LINUX终端后会出现Shell提示符，等待用户输入命令。Bash、B Shell和K Shell的缺省提示符为“$”，C Shell的缺省提示符为“%”。当以root身份登录时，系统提示符为“#”。

命令格式

在Shell提示符下，就可以输入LINUX命令。LINUX命令的基本格式如下：

command 参数1 参数2 ... 参数n

LINUX命令由一个命令(command)和零到多个参数构成，命令和参数之间，以及参数与参数之间用空格隔开。 LINUX的命令格式和DOS的命令格式相似，但LINUX命令区分大小写，且命令和参数之间必须隔开。

如：对DOS来说是以下四条命令是相同的：

cd\tmp

cd \tmp

CD \tmp

cd \Tmp

UNIX命令可分为两类：

。Shell内部命令----包括简单常用命令，随shell 装入内存

。Shell 外部命令----不常驻内存，用时调入。

对LINUX操作系统来说，改变目录的命令也是cd命令，以下是正确的LINUX cd命令：但tmp和Tmp是两个不同的目录。

cd /tmp

cd /Tmp

以下两条命令已经不是cd命令了。

cd/tmp

CD/tmp

LINUX常用简单命令

解释	LINUX命令	命令举例
帮助	man	man date
查看日期	date	date
显示日历	cal	cal 1998
显示大字	banner	banner "ABCD"
计算器	bc	bc
修改口令	passwd	passwd
查看谁在使用	who	who
查看我是谁	who am i	who am i
清除屏幕	clear	clear

注销（退出系统）

在每次使用完后，一定要进行注销，以防他人通过你的帐号进入系统，并保证系统的完整性。注销过程如下：在系统提示符下，运行：

```
$ exit
```

或

```
$ logout
```

或直接按键：

Ctrl+d。

由于系统的不同，注销的命令也可能不同。

第二章 文件系统

2.1 LINUX文件系统分类

LINUX操作系统可由多个可以动态安装及拆卸的文件系统组成。 LINUX文件系统主要分为两大类：根文件系统和附加文件系统。

根文件系统（the root file system） 每一个LINUX操作系统在其主硬盘上至少含有一个文件系统，它包含构成操作系统的程序和目录，一般由符号“/”来表示。

附加文件系统 除根文件系统外的其它文件系统，如/u文件系统，AFS文件系统等。

以后如无特别声明，都是针对根文件系统来说明的。

2.2 文件类型

在LINUX中文件共分为四种：

1) 一般文件(ordinary file),分为:	(1) 文本文件 (2) 二进制文件
2) 目录文件(directory)	
3) 特殊文件(special file)，分为:	1 块设备文件 2 字符设备文件
4) 符号链接文件(symbolic links)	

文件类型：

-	普通文件
d	目录文件
b	块设备文件
c	字符设备文件
p	管道设备
l	符号链接文件

举例：

$ ls -l

-rwxr-xr-- 2 wjm newservice 321 Oct 17 09:33 file1

drwxr-xr-x 2 wjm newservice 96 Oct 17 09:40 dir1

2.3 目录结构

LINUX操作系统采用树型带勾连的目录结构。在这种结构中，一个文件的名字是由根目录到该文件的路径上的所有节点名按顺序构成的，相互之间用“/”分开。如文件prog的全路径名为：/usr/smith/prog，根目录用“/”表示。

根文件系统的常用目录举例如下表：

/bin	大部分可执行的UNIX命令和共用程序
/dev	设备文件，如/dev/cd0
/etc	系统管理命令和数据文件
/lib	C程序库
/usr	存放用户的家目录和用户共用程序或文件
/tmp	临时工作目录，存放一些临时文件

家目录 LINUX在创建用户名时，自动在/home目录下创建与用户名同名的子目录，如/home/smith子目录，这个子目录成为此用户的家目录（Home Directory）。家目录中有一个文件比较特殊：“.profile”(或“.login”)文件。当以该家目录的用户名登录LINUX时，会自动执行“.profile”文件。它有点类似于DOS的AUTOEXEC.BAT文件。“.profile”是B/K shell的启动文件，而“.login”是C shell的启动文件。“.profile”文件中常有一行：

PATH=$HOME:$HOME/bin

其中，PATH类似DOS的PATH,而$HOME表示家目录。

2.4 文件名称

LINUX文件名称：

大小写敏感（file1和File1表示两个不同的文件）；

UNIX无专用扩展名(文件名可出现多个点，并无特殊含义)。

特殊文件名及其通配符：

/	表示根目录
.	表示当前目录
..	表示当前目录的父目录
.字符串	表示隐含文件，如".profile"文件

?	任一个字符，如"abc?"
*	任何字符，如"ab*10"表示以ab开头和以10结尾的名字

2.5 文件存取权限

LINUX文件有三种存取权限：

权限	普通文件的存取权限	目录的存取权限
r	具有读取文件的权利	能读取文件名称
w	具有写入文件的权利	能建立和删除文件，可以改变文件名
x	具有执行文件的权利	能使用该目录下的文件(如cd命令)搜索文件等

有三种类型的用户可以存取文件：

用户类型	说明
owner	文件的拥有者
group	文件所在的工作组
other	其他用户(非owner和非group)

每种类型的用户都有三种文件存取权限：r、w、x。

文件存取权限的显示可以通过“ls -l”命令显示，如：

$ ls -l file1

显示如下：

-rwxr-xr-- 2 wjm newservice 321 Oct 17 09:33 file1

用chmod命令修改文件的存取权限：

chmod symbolic_mode file..

用户类型简写

u	**拥有者(user --- owner)**
g	**与拥有者同一组的用户(group)**
o	**其他人(other)**
a	**所有人(all)**

改变文件属主的格式如下：(必须对文件具有write权利才可改变文件的拥有者)

chown 属主名 文件名

chown wjm test1

改变文件所在组的格式如下：(必须是文件的拥有者或supervisor才能改变文件组别的归属)

chgrp 组名 文件名

chgrp newservice test1

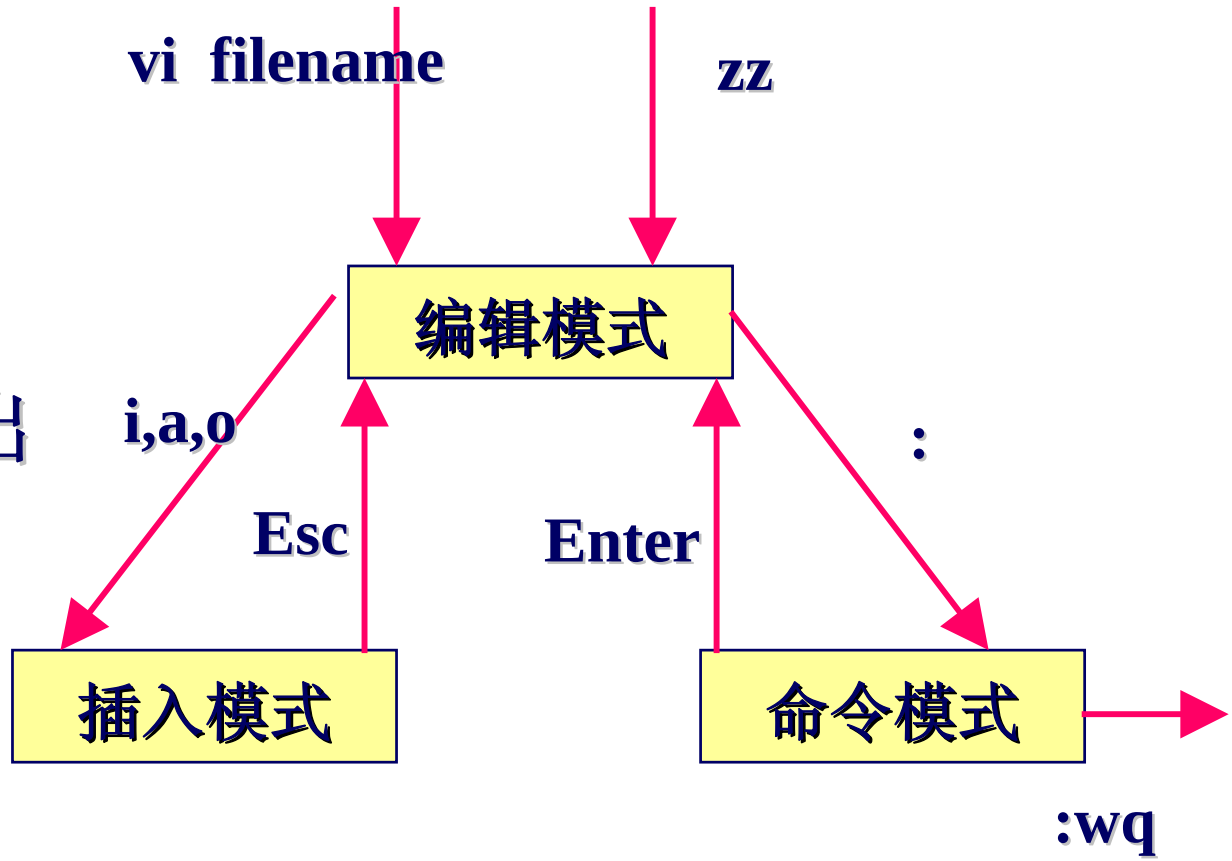
2.6 LINUX文件系统常用命令

	命令	命令举例	类似DOS命令
显示当前目录	pwd	pwd	cd
改变目录	cd	cd /usr	cd c:\usr
进入家目录	cd	cd	
创建目录	mkdir	mkdir abc	md abc
删除空目录	rmdir	rmdir abc	rd abc
删除目录及其内容	rm -r	rm -r abc	deltree abc
显示目录内容	ls	ls -l abc(文件长列表)ls -a abc(所有类型文件)	dir abc
显示文本文件内容	cat	cat file1.c	type file1.c
分屏显示文本内容	more	more file1.c	
拷贝文件	cp	cp file1 file2	copy file1 file2
移动(重命名)文件	mv	mv call.test call.list	move call.tst call.lst ren call.test bbb
删除文件	rm	rm call.list	del call.lst

第三章 文件系统常用工具及其深入了解

一、Vi (visual interpreter)编辑器

- vi的三种模式:
- 编辑模式
- 插入模式
- 命令模式
- 存盘与退出



操作符	操作效果
h(←)	光标左移一个字符
l(→)	光标右移一个字符
k(↑)	光标上移一行
j(↓)	光标下移一行
G	光标移至文件最后一行
0	光标移至首行
$	光标移至行尾
H	光标移至屏幕的最上行
M	光标移至屏幕的中部
L	光标移至屏幕的最下行
w	光标右移一个单词

编辑模式

替换和删除

- rc-用c替换光标所指向的当前字符
- x-删除光标所指向的当前字符
- dd-删除光标所在行，并去除空隙。

- 粘贴和复制
  - yy-复制当前行到内存缓冲区。
  - P-将缓冲区的内容粘贴到光标的前（后）面。

编辑模式

- 搜索字符串
  - /str-正向搜索字符串str.
  - n-继续搜索，找出str下次出现的位置。
  - N-在光标处向反方向重复前一搜索命令
  - ？ str-反向搜索字符串str.
- 撤销和重复
  - u -撤消前一条命令的结果
  - . -重复最后一条修改正文的命令。

插入模式

- 进入插入模式(INSERT提示)
  - i - 在光标左侧输入正文
  - a -在光标右侧输入正文
  - I -在光标所在行的开头输入正文
  - A-在光标所在行的末尾输入正文
- 正文替换
  - s-用输入的正文替换光标所指向的字符。

命令模式

- 在命令方式下运行。不管在什么状态，最好在运行下面命令前，先按一下ESC键，以防出错。
- 退出命令
  - :q-在未作修改的情况下退出
  - :q-放弃所有修改，退出编辑程序。
- 行号与文件
  - :n-将光标移到第n行
  - :w-将编辑的内容写入原始文件，用来保存中间结果

命令模式

- 文件
  - :wq-将编辑的内容写入原始文件并退出编辑程序
  - :w file -将编辑的内容写入file文件，保持原有内容不变。
  - : f file -将当前文件重新命名为file。

  :w >>file　将内容写至文件原有内容之后

  :w! file　强行进行写盘文件**file**的动作

命令模式

- 字符串搜索
  - :/str/ -正向搜索，将光标移动到下一个包含字符串str的行。
  - :/str1/,/str2/w file -正向搜索，并将包含字符串str1的行至包含字符串str2的行写入file文件。

命令模式

- 正文替换
  - : s/str1/str2/ -用字符串str2替换行中首次出现的字符串str1
  - : s/str1/str2/g -用字符串str2替换行中所有出现的字符串str1
- 删除正文
  - :d- 删除光标所在行
  - : /str1/,/str2/d -删除从字符串str1到str2的所有行

使用vi中应注意事项

1) 对vi多种方式使用不习惯,插入方式与命令方式切换出现混乱。查对当前方式,可用[Esc]

2) 文档编辑时,屏幕产生混乱

屏幕刷新有误,用ctrl-l/ctrl-r重新刷新

3) 对屏幕中显示信息进行操作时没反映

可能屏幕输出进程被挂起,用ctrl-q解脱

4) 编辑完成后不能正确退出vi

系统有问题,应先存盘(:w newfile),利用/tmp目录

二、利用find命令查找文件

find命令会在指定目录及其子目录下查找符合条件的特定文件。此命令的最大用处是当忘了文件的正确所在，而想找到该文件。

命令格式：　　**find**　目录名　条件

目录名 --- 欲开始寻找的目录所在。**find**会寻找此目录及其子目录。可以有多个目录名称，只要目录与目录之间用空格分开即可。

条件　--- 欲搜索文件的条件，可包含文件名称、拥有者、最后修改时间等等。

条件列表说明：

选项：- name filename 查找名为filename的文件

-type x 查找类型为x的文件，x=d目录 x=f普通文件

-user username 查找属主为username的文件

-size n 查找记录块为n的文件

-atime n 查找n天前被访问过的文件

-mtime n 查找n天前被修改过的文件

-exec cmd{ } 对符合条件的文件执行cmd命令，{ }表示所找到的文件，命令要用“\;”结束

-a 选择同时满足多个条件的文件（逻辑与）

-o 选择满足若干条件之一的文件（逻辑或）

（expr) 括号中的表达式作为整体判断

-print 显示找到文件的路径名

例： % find /home \(-name core -o -name a.out \)

-atime +3 -exec rm { }\;

% find / -name “y2k*” -print

grep :按给定的模式搜索文件

格式：**grep [options] pattern file**

选项：**-v** 显示所有不匹配该模式的行

-c 仅显示匹配该模式的行

-l 仅显示包含指定模式的文件名

-y 模式不区分大小写

例：**%grep the tempfile** 显示文件**tempfile**中含字符串 **the**的行

%grep -v the tempfile 显示文件**tempfile**中不含字符串 **the**的行

另：**grep** 命令的两个变种命令：

fgrep :只搜索字符串不能匹配正则表达式，是固定快速形式

如 % ls -l ｜fgrep ‘^d’是错误的命令。

egrep :支持完全正则表达式搜索及快速算法,是**grep**的扩展形式

三、利用cmp命令比较文件

cmp可比较两个文件，这两个文件可以是文本文件也可以是非文本文件。cmp命令会显示两个文件第一次数据不同的位置。若两个文件内容相同，则不显示任何信息。

命令格式：cmp [-ls] 文件名1 文件名2

-l 显示每个不同之处不同字节的内容

-s 只显示cmp命令的返回值（相同为0，不相同为1）

四、df:报告磁盘空间使用情况

格式：df [options] [resource]

选项：- i (BSD用）报告已用和尚未用的 i节点数

-k 以k字节为单位显示磁盘使用信息

-F type (systv)报告指定类型的文件系统使用情况，其中type 可以是nfs , ufs 等值(BSD的是-t type)

例：sun solaris 2.5系统中用：%df 可显示如下：

文件系统名	块设备名	剩余块数	文件数
/	（dev/dsk/cot3d0s0):	110390 blocks	72540 files
/usr	（dev/dsk/cot3d0s6):	140116 blocks	290492files
/proc	(/proc):	0 blocks	938 files

五、file :判断文件类型

格式：file filename

此命令可推测出文件类型，显示方式为：

test.c： c program text

net：symbolic link to /net/sun4/net (符号连接）

brie：executuble shell script（shell 脚本文件）

lib：directory（目录）等等

六、ps :显示当前进程状态

格式：ps [options] [namelist]

选项：-e 显示当前运行的每一个进程的信息

-f 产生一个父子关联进程完整的清单

-l 产生一个长清单

例：% ps

显示：pid tty time command

七、在后台运行程序

要让程序在后台执行，只需在命令行的最后加上“&”符号。

[例1] 在后台运行find命令，在当前目录及其子目录下查找文件名为abc的文件。

执行命令和立即显示的内容如下：

```
$ find . -name  abc  -print&
10722
$
```

10722表示进程号（PID）。当find命令在后台执行完后，会显示结果。

当在后台运行命令时，最好将其输出重定向到一个文件中去，以便以后检查。

八、在指定时间执行指定程序

本节介绍cron和crontab。

cron是一个系统进程，在UNIX的多用户模式下执行。它使某程序在一特定的时间被执行。cron会按照crontabs文件所指定的时间执行命令，crontabs文件内容举例如下：

```
#min  hour    day     month      day_of_week    command
#0-59   0-23   1-31     1-12       0-6(Sunday=0)
#-------------------------------------------------
30        0       *       *              *              /usr/abc/rj
0         7       *    1,3-6,8-12        1-5        /usr/abc/makfile
```

九、如何控制后台进程

1、显示进程信息

LINUX命令	解释
ps	显示使用终端有关的进程信息
ps -u 用户名	显示某用户的进程（如ps -u abc）
ps -e	显示所有正在运行的进程信息
ps -f	长列表显示每个进程信息

2、用kill命令杀掉后台进程

UNIX命令	解释
kill 进程号	删除进程
kill -9 进程号	强制删除进程

3、停止失控进程步骤

(1) 在未锁死的终端以root登录
(2) 用"ps -ef"和"grep 关键字"找到失控进程pid号
(3) kill pid号，若杀不掉，再运行"kill -9 pid号"
(4) 若还杀不掉，则shutdown

第四章 LINUX用户管理及实用程序

LINUX是多用户系统，它应能完成：

。多个用户能共享系统资源

。多个用户间要有良好的安全保密措施

帐户作用：保存系统辨认用户身份的信息，

记录并组织用户在系统中的活动。

相关的文件：

文件**/etc/passwd** ——存储帐户管理信息，被授权的用户占有一项记录信息，其信息格式如下：

username : x : uid : gid : comment : homedir : loginshell

(1) (2) (3) (4) (5) (6) (7)

例如：

zhang:*:103:100:zhang hongguang:/home/zhang:/bin/csh

（1）：用户名

（2）：口令，加密显示

（3）：用户标识

（4）：用户组标识

（5）：用户注册描述

（6）：用户主目录

（7）：进入系统后第一步完成的工作

系统中设置两类用户帐户：

- 超级用户帐户---拥有系统最高权限，分配给系统管理员有特定的用户名“root”及相应的口令；
- 普通用户帐户---由系统管理员分配的用户权限，给各种用户，用于管理自己的文件和目录。

用户口令管理

文件/etc/shadow：存储帐户中用户加密后的口令及管理信息

格式：

username:password:lastchg:min:max:warn:inactive:expire

(1) (2) (3) (4) (5) (6) (7) (8)

（1）：用户名

（2）：加密后的口令

（3）：口令最后修改日期

（4）：可修改口令的最少天数

（5）：可修改口令的最多天数

（6）：口令到期提前提示天数

（7）：允许不登录天数

（8）：口令到期确定日期

用户组信息

文件/etc/group——存储与用户帐户相关的组信息

格式：

groupname : password : gid : user-list

(1) (2) (3) (4)

（1）：用户组名称

（2）：用户组口令

（3）：用户组标识

（4）：组中的用户列表

第五章　shell的输入与输出

(1) 在shell脚本中，可以用几种不同的方式读入数据：可以使用标准输入—缺省为键盘，或者指定一个文件作为输入。

(2) 对于输出也是一样：如果不指定某个文件作为输出，标准输出总是和终端屏幕相关联。

(3) 如果所使用命令出现了什么错误，它也会缺省输出到屏幕上，如果不想把这些信息输出到屏幕上，也可以把这些信息指定到一个文件中。

本章我们将讨论以下内容:

- 使用标准输入、标准输出及标准错误。
- 重定向标准输入和标准输出。

5.1 echo

使用**echo**命令可以显示文本行或变量，或者把字符串输入到文件。它的一般形式为:

echo string

echo命令有很多功能，其中最常用的是下面几个:

\c 不换行。

\f 进纸。

\t 跳格。

\n 换行。

如果希望提示符出现在输出的字符串之后，可以用:

```
$ echo  "What is your name :\c"
$ read name
```

上面的命令将会有如下的显示:

```
What is your name :□
```

其中“□”是光标。

如果想在输出字符之后，让光标移到下一行，可以用:

```
$ echo "The red pen ran out of ink"
```

如果是LINUX系统，那么.....

必须使用- n选项来禁止echo命令输出后换行:

```
$ echo  -n  "What is your name :"
```

初涉shell的用户常常会遇到的一个问题就是如何把双引号包含到echo命令的字符串中。

引号是一个特殊字符，所以必须要使用反斜杠 \ 来使shell忽略它的特殊含义。假设你希望使用echo命令输出这样的字符串:

“/ dev / rmt0”

那么我们只要在引号前面加上反斜杠\即可:

$ echo "\"/dev/rmt0"\"

" /dev/rmt0 "

如果希望在**echo**命令输出之后附加换行，可以使用 **\n** 选项:

```
$ pg echod
#!/bin/sh
echo  "this echo's 3 new lines\n\n\n"
echo "OK"
```

运行时会出现如下输出:

```
$ echod
this echo's 3 blank lines



OK
```

还可以在**echo**语句中使用跳格符，记住别忘了加反斜杠\:

```
$ echo "here is a tab\there are two tabs\t\tok"
here is a tab   here are two tabs                ok
```

如果是**LINUX**系统，那么...

别忘了使用 **–e** 选项才能使转义符生效:

```
$ echo -e "here is a tab\there are two tabs\t\tok"
here is a tab   here are two tabs                ok
```

5.2 read

可以使用**read**语句从键盘或文件的某一行文本中读入信息，并将其赋给一个变量。如果只指定了一个变量，那么**read**将会把所有的输入赋给该变量，直至遇到第一个文件结束符或回车。

它的一般形式为:

read varible1 varible2 ...

在下面的例子中，只指定了一个变量，它将被赋予直至回车之前的所有内容:

```
$ read name
Hello I am superman
$ echo $name
Hello I am superman
```

在下面的例子中，我们给出了两个变量，它们分别被赋予名字和姓氏。**shell**将用空格作为变量之间的分隔符:

```
$ read name surname
John Doe
$ echo $name $surname
John Doe
```

如果输入文本域过长， Shell 将所有的超长部分赋予最后一个变量。下面的例子，假定要读取变量名字和姓，但这次输入三个名字；结果如下：

```
$ read name surname
John Lemon Doe
$ echo $name
John
$ echo $surname
Lemon Doe
```

在编写**shell**脚本的时候，如果担心用户会对此感到迷惑，可以采用每一个**read**语句只给一个变量赋值的办法:

```
$ pg var_test
#!/bin/sh
# var_test
echo "First Name :\c"
read name
echo "Middle Name :\c"
read middle
echo "Last name :\c"
read surname
```

5.3 cat

cat是一个简单而通用的命令，可以用它来显示文件内容，创建文件，还可以用它来显示控制字符。在使用cat命令时要注意，它不会在文件分页符处停下来；它会一下显示完整个文件。如果希望每次显示一页，可以使用more命令或把cat命令的输出通过管道传递到另外一个具有分页功能的命令中，请看下面的例子:

```
$ cat myfile | more
```

或

```
$ cat myfile | pg
```

cat命令的一般形式为:

cat [options] filename1 ... filename2 ...

例: $ cat myfile

$ cat myfile1 myfile2 myfile3

$ cat myfile1 myfile2 myfile3 > bigfile

```
$ cat > myfile
This is great
<CTRL-D>
$ pg myfile
This is great
```

还可以使用**cat**命令来显示控制字符。

这里有一个对从**DOS**机器上**ftp**过来的文件进行检察的例子，在这个例子中，所有的控制字符< **CTRL-M** >都在行末显示了出来。

```
$ cat -v life.tct
ERROR ON REC AS12^M
ERROR ON REC AS31^M
```

5.4 管道

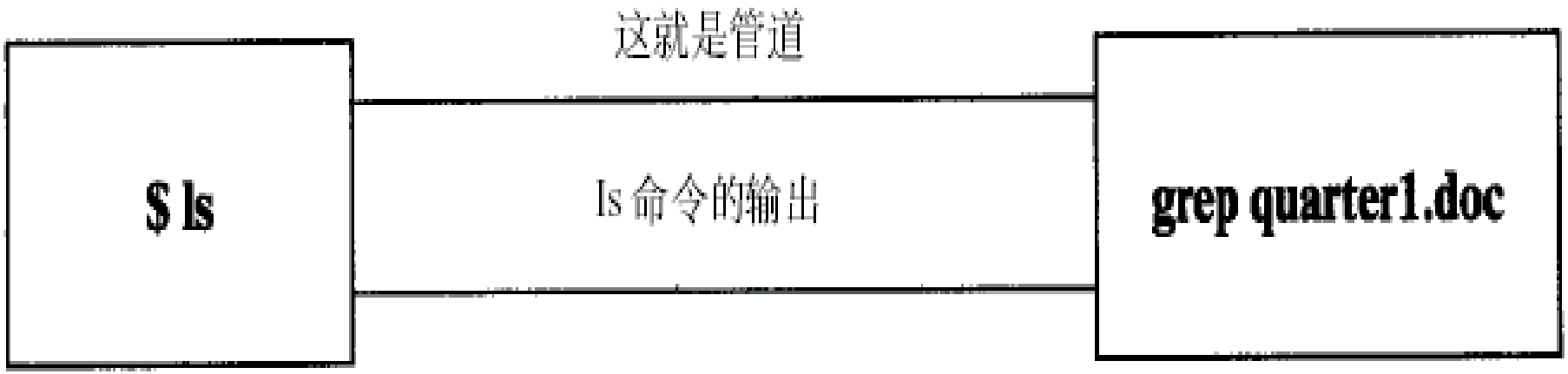
可以通过管道把一个命令的输出传递给另一个命令作为输入。管道用竖杠|表示。它的一般形式为:

命令1 | 命令2

其中 | 是管道符号。

在下面的例子中，在当前目录中执行文件列表操作，如果没有管道的话，所有文件就会显示出来。当**shell**看到管道符号以后，就会把所有列出的文件交给管道右边的命令。

```
$ ls | grep quarter1.doc
quarter1.doc
```




```
$ ls        accounts.doc   acc_LPSO.doc        quarter1.doc
            quarter1.doc   quarter2.doc
```

$ sort myfile | lp

```
$ df -k | awk '{print $1}'| grep -v "Filesystem"
/dev/hda5
/dev/hda8
/dev/hda6
/dev/hdb5
/dev/hdb1
/dev/hda7
/dev/hda1
```

使用管道把df命令的输出传递给awk命令，awk显示出其中的第一列。你还可以再次使用管道把awk的结果传递给grep命令，去掉最上面的题头filesystem。

5.5 tee

tee命令作用可以用字母T来形象地表示。它把输出的一个副本输送到标准输出，另一个副本拷贝到相应的文件中。如果希望在看到输出的同时，也将其存入一个文件，那么这个命令再合适不过了。

它的一般形式为:

tee -a files

其中，-a表示追加到文件末尾。

```
$ who | tee who.out
louise    pts/1     May 20 12:58 (193.132.90.9)
matthew   pts/0     May 20 10:18 (193.132.90.1)

cat who.out
louise    pts/1     May 20 12:58 (193.132.90.9)
matthew   pts/0     May 20 10:18 (193.132.90.1)
```

如果不想把输出重定向到文件中，可以不这样做，而是把它定向到某个终端上。在下面的例子中，一个警告被发送到系统控制台上，表明一个磁盘清理进程即将运行。

```
$ echo "stand-by disk cleanup

  starting  in 1 minute" |  tee /dev/console
```

5.6 标准输入、输出和错误

当我们在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。由于文件描述符不容易记忆， shell同时也给出了相应的文件名。

下面就是这些文件描述符及它们通常所对应的文件名:

文件	文件描述符
输入文件—标准输入	0
输出文件—标准输出	1
错误输出文件—标准错误	2

系统中实际上有1 2个文件描述符，但是正如我们在上表中所看到的， 0、1、2是标准输入、输出和错误。可以任意使用文件描述符3到9。

5.6.1 标准输入

标准输入是文件描述符0。它是命令的输入，缺省是键盘，也可以是文件或其他命令的输出。

5.6.2 标准输出

标准输出是文件描述符1。它是命令的输出，缺省是屏幕，也可以是文件。

5.6.3 标准错误

标准错误是文件描述符2。这是命令错误的输出，缺省是屏幕，同样也可以是文件。你可能会问，为什么会有一个专门针对错误的特殊文件？这是由于很多人喜欢把错误单独保存到一个文件中，特别是在处理大的数据文件时，可能会产生很多错误。

5.7 文件重定向

在执行命令时，可以指定命令的标准输入、输出和错误，要实现这一点就需要使用文件重定向。在对标准错误进行重定向时，必须要使用文件描述符，但是对于标准输入和输出来说，这不是必需的。

command > filename 把标准输出重定向到一个新文件中

command >> filename 把标准输出重定向到一个文件中(追加)

command 1 > fielname 把标准输出重定向到一个文件中

command > filename 2>&1 把标准输出和标准错误一起重定向到一个文件中

command 2 > filename 把标准错误重定向到一个文件中

command 2 >> filename 把标准输出重定向到一个文件中(追加)

command >> filename 2>&1 把标准输出和标准错误一起重定向到一个文件中(追加)

command < filename >filename2 把command命令以filename文件作为标准输入，以filename2文件作为标准输出

command < filename command命令以filename文件作为标准输入

command << delimiter 把从标准输入中读入，直至遇到delimiter分界符

5.7.1 重定向标准输出

```
$ ls -l | grep ^d >>files.out
$ ls account* >> files.out
```

在上面的例子中，所有的目录名和以**account**开头的文件名都被写入到**file.out**文件中。

如果想创建一个长度为**0**的空文件，可以用'**> filename** ':

$ > myfile

5.7.2 重定向标准输入

$ sort < name.txt ⟺ $ sort name.txt

$ sort < name.txt > name.out

$ mail louise < contents.txt

重定向操作符**command << delimiter**是一种非常有用的命令，通常都被称为“此处”文挡。

shell将分界符**delimiter**之后直至下一个同样的分界符之前的所有内容都作为输入，遇到下一个分界符，**shell**就知道输入结束了。

```
$ cat >> myfile <<MAYDAY
> Hello there I am using a $TERM terminal
> and my user name is $LOGNAME
> bye...
> MAYDAY

$ pg myfile
Hello there I am using a vt100 terminal
and my user name is dave
bye...
```

可以任意定义分界符**delimiter**，最常见的是EOF，上例使用MAYDAY，这完全取决于个人的喜好。

5.7.3 重定向标准错误

为了重定向标准错误，可以指定文件描述符2。如果你在对更重要的文件进行操作，可能会希望保存相应的错误。

$ grep "trident“ missiles 2 > /dev/null

这样所有的错误输出都输送到了/ dev / null，不再出现在屏幕上。

```
$ grep "LPSO" * 2>>account.err
$ grep "SILO" * 2>>account.err
```

5.8 结合使用标准输出和标准错误

一个快速发现错误的方法就是，先将输出重定向到一个文件中，然后再把标准错误重定向到另外一个文件中。下面给出一个例子:

有两个审计文件，其中一个的确存在，而且包含一些信息，而另一个由于某种原因已经不存在了(但我不知道)。我想把这两个文件合并到**accounts.out**文件中。

$ cat account_qtr.doc account_end.doc

1> accounts.out 2> accounts.err

现在如果出现了错误，相应的错误将会保存在**accounts.err**文件中。

```
$ pg accounts.out
AVBD 34HJ    OUT
AVFJ    31KO  OUT
...

$ pg accounts.err
cat: account_end.doc: No such file or directory
```

事先并不知道是否存在**account_end.doc**文件，使用上面的方法能够快速发现其中的错。

5.9 合并标准输出和标准错误

在合并标准输出和标准错误的时候，切记shell是从左至右分析相应的命令的。下面给出一个例子:

```
$ cleanup  >  cleanup.out   2>  &1
```

在上面的例子中，我们将cleanup脚本的输出重定向到cleanup.out文件中，而且其错误也被重定向到相同的文件中。

```
$ grep  "standard "  *   >  grep.out   2>  &1
```

在上面的例子中， grep命令的标准输出和标准错误都被重定向到grep.out文件中。

第六章　命令执行顺序

在执行某个命令的时候，有时需要依赖于前一个命令是否执行成功。例如，假设你希望将一个目录中的文件全部拷贝到另外一个目录中后，然后删除源目录中的全部文件。在删除之前，你希望能够确信拷贝成功，否则就有可能丢失所有的文件。

如果希望在成功地执行一个命令之后再执行另一个命令，或者在一个命令失败后再执行另一个命令，**&&** 和 **||** 可以完成这样的功能。相应的命令可以是系统命令或**shell**脚本。

Shell还提供了在当前**shell**或子**shell**中执行一组命令的方法，即使用（ ）和**{ }**。

6.1 使用&&

使用&&的一般形式为:

命令1 && 命令2

这种命令执行方式相当地直接。&&左边的命令（命令1）返回真(即返回0，成功被执行）后，&&右边的命令（命令2）才能够被执行；换句话说，“如果这个命令执行成功&&那么执行这个命令”。

这里有一个使用&&的简单例子:

```
$ mv   /apps/bin   /apps/dev/bin
  &&   rm   -r   /apps/bin
```

```
$ sort  quarter_end.txt  >
  quarter.sorted  &&  lp quarter.sorted
```

6.2 使用||

使用||的一般形式为:

命令1 || 命令2

||的作用有一些不同。如果||左边的命令（命令1）未执行成功，那么就执行||右边的命令（命令2）；或者换句话说，“如果这个命令执行失败了|| 那么就执行这个命令”。

$ cp a.txt b.txt || echo " If you are

^ seeing this cp failed"

cp : a.txt : No such file or directory

If you are seeing this cp failed

$ comet month_end.txt || exit

首先对month_end.txt文件执行了一个名为comet的shell脚本，如果该脚本未执行成功，该shell将结束。

在编写shell脚本时，使用&&和||对构造判断语句非常有用。如果希望在前一个命令执行失败的情况不执行后面的命令，那么以上所讲述的方法非常简单有效。使用这样的方法，可以根据&&或||前面命令的返回值来控制其后面命令的执行。

6.3 用（ ）将命令结合在一起

如果希望把几个命令合在一起执行， shell提供了两种方法。既可以在当前shell也可以在子shell中执行一组命令。

为了在当前shell中执行一组命令，可以用命令分隔符隔开每一个命令，并把所有的命令用圆括号（ ）括起来。

它的一般形式为:

（命令1; 命令2; ...）

很少单独使用这种方法。一般只和&&或||一起使用这两种方法。

```
$ sort  a.txt  > b.sorted  && (cp
  ^b.sorted   /logs/b.sorted ;  lp
  b.sorted )
```

使用命令组合的方法，如果**sort**命令执行成功了，可以先将输出文件拷贝到一个日志区，然后再打印。

第七章　正则表达式介绍

随着对**UNIX**和**LINUX**熟悉程度的不断加深，需要经常接触到正则表达式这个领域。使用**shell**时，从一个文件中抽取多于一个字符串将会很麻烦。例如，在一个文本中抽取一个词，它的头两个字符是大写的，后面紧跟四个数字。如果不使用某种正则表达式，在**shell**中将不能实现这个操作。

正则表达式是一些特殊或不很特殊的字符串模式的集合。

为了抽取或获得信息，我们给出抽取操作应遵守的一些规则。这些规则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进行搜寻。例如，**A**将查询**A**，**x**将查找字母**x**。

系统自带的所有大的文本过滤工具在某种模式下都支持正则表达式的使用。

7.1 使用句点匹配单字符

句点“.”可以匹配任意单字符。例如，如果要匹配一个字符串，以**beg**开头，中间夹一个任意字符，那么可以表示为**beg.n**，“.”可以匹配字符串头，也可以是中间任意字符。

在ls -l命令中，可以匹配一定权限:

. . . x . . x . . x

此格式匹配用户本身，用户组及其他组成员的执行权限。

drwxrwxrw-　　– no match

-rw-rw-rw-　　– no match

-rwx-rwxr-x　　– match

-rwx-r-x-r-x　　– match

假定正在过滤一个文本文件，对于一个有10个字符的脚本集，要求前4个字符之后为XC，匹配操作如下：

....XC....

以上例子解释为前4个字符任意，5，6字符为XC，后4个字符也任意，按下例运行：

1234XC9088 – match

4523XX9001 – no match

0011XA9912 – no match

9931XC3445 – match

注意：“.”允许匹配ASCII集中任意字符，或为字母，或为数字。

7.2 在行首以^匹配字符串或字符序列

^只允许在一行的开始匹配字符或单词。例如，使用 ls –l 命令，并匹配目录。之所以可以这样做是因为 ls –l 命令结果每行第一个字符是d，即代表一个目录。

```
^d
drwxrwxrw-        – match
-rw-rw-rw-        – no match
drwx-rwxr-x       – match
-rwx-r-x-r-x      – no match
```

在行首第4个字符为1，匹配操作表示为：

^...1

结果为：

1234XC9088 – no match

4523XX9001 – no match

0011XA9912 – match

9931XC3445 – match

7.3 在行尾以$匹配字符串或字符

可以说$与^正相反，它在行尾匹配字符串或字符， $符号放在匹配单词后。假定要匹配以单词**trouble**结尾的所有行，操作为：

trouble$

类似的，使用**1d$**返回每行以**1d**结尾的所有字符串。如果要匹配所有空行，执行以下操作：

^ $

具体分析为匹配行首，又匹配行尾，中间没有任何模式，因此为空行。

7.4 使用*匹配字符串中的单字符或其重复序列

使用此特殊字符匹配任意字符或字符串的重复多次表达式。例如:

compu*t

将匹配字符**u**一次或多次:

computer

computing

compuuuuute

7.5 使用 \ 屏蔽一个特殊字符的含义

有时需要查找一些字符或字符串，而它们包含了系统指定为特殊字符的一个字符。什么是特殊字符？一般意义上讲，下列字符可以认为是特殊字符：

$. ' " * [ ] ^ | ( ) \ + ?

假定要匹配包含字符“ .”的各行而“.”代表匹配任意单字符的特殊字符，因此需要屏蔽其含义。操作如下：

\.

上述模式不认为反斜杠后面的字符是特殊字符，而是一个普通字符，即句点。

7.6 使用[ ]匹配一个范围或集合

使用[ ]匹配特定字符串或字符串集，可以用逗号将括弧内要匹配的不同字符串分开，但并不强制要求这样做（一些系统提倡在复杂的表达式中使用逗号），这样做可以增加模式的可读性。

使用“ -”表示一个字符串范围，表明字符串范围从“ -”左边字符开始，到“ -”右边字符结束。

如果熟知一个字符串匹配操作，应经常使用[ ]模式。

假定要匹配任意一个数字，可以使用:

[0123456789]

然而，通过使用“-”符号可以简化操作:

[0-9]

如要匹配任意字母或数字，模式如下:

[A-Za-z0-9]

如要匹配Computer或computer两个单词，
可做如下操作:

[Cc]omputer

为抽取诸如Scout、shout、bought等单词，
使用下列表达式:

[ou] .*t

注意^符号的使用，当直接用在第一个括号里，
意指否定或不匹配括号里内容。

[^a-zA-Z]

匹配任一非字母型字符，而

[^ 0 - 9]

匹配任一非数字型字符。

7.7 使用 \{\} 匹配模式结果出现的次数

使用*可匹配所有匹配结果任意次，但如果只要指定次数，就应使用\{\}，此模式有三种形式，即：

pattern\{n\} 匹配模式出现n次。

pattern\{n,\} 匹配模式出现最少n次。

pattern\{n,m\} 匹配模式出现n到m次之间，n，m为0-255中任意整数。

请看第一个例子，匹配字母A出现两次，并以B结尾，操作如下：

A\{2\}B

匹配值为**A A B**

Eg2: 匹配A至少4次，使用:

A\{4,\}B

可以得结果**AAAAB**或**AAAAAAAB**，但不能为**AAAB**。

Eg3: 如给出出现次数范围，例如A出现2次到4次之间:

A\{2,4\}B

则结果为**AAB**、**AAAB**、**AAAAB**，而不是**AB**或**AAAAAB**等。

假定从下述列表中抽取代码:

1234XC9088
4523XX9001
0011XA9912
9931XC3445

格式如下：前4个字符是数字，接下来是xx，最后4个也是数字，操作如下：

[0-9]\{4\}X X[0-9]\{4\}

1234XC9088 – no match
4523XX9001 – match
0011XA9912 – no match
9931XC3445 – no match

在写正则表达式时，可能会有点难度或达不到预期效果，一个好习惯是在写真正的正则表达式前先写下预期的输出结果。这样做，当写错时，可以逐渐修改，以消除意外结果，直至返回正确值。

在**shell**编程中，一段好的脚本与完美的脚本间的差别之一，就是要熟知正则表达式并学会使用它们。相比较起来，用一个命令抽取一段文本比用三四个命令得出同样的结果要节省许多时间。

第八章　tr 用法

tr用来从标准输入中通过替换或删除操作进行字符转换。tr主要用于删除文件中控制字符或进行字符转换。使用tr时要转换两个字符串：字符串1用于查询，字符串2用于处理各种转换。

tr刚执行时，字符串1中的字符被映射到字符串2中的字符，然后转换操作开始。

本章内容有:

- 大小写转换。
- 去除控制字符。
- 删除空行。

带有最常用选项的**tr**命令格式为:

tr –c -d -s [字符串1] [字符串2] file

这里:

-c 用字符串**1**中字符集的补集替换此字符集，要求字符集为**ASCII**。

-d 删除字符串**1**中所有输入字符。

-s 删除所有重复出现字符序列，只保留第一个；即将重复出现字符串压缩为一个字符

file是转换文件名。虽然可以使用其他格式输入，但这种格式最常用。

8.1.1 字符范围

使用tr时，可以指定字符串列表或范围作为形成字符串的模式。这看起来很像正则表达式，但实际上不是。指定字符串1或字符串2的内容时，只能使用单字符或字符串范围或列表。

像大多数系统工具一样， tr也受特定字符的影响。因此如果要匹配这些字符，需使用反斜线屏蔽其特殊含义。例如，用\{指定花括号左边可以屏蔽其特殊含义。

速记符	含义	八进制方式
\a	Ctrl-G 铃声	\007
\b	Ctrl-H 退格符	\010
\f	Ctrl-L 走行换页	\014
\n	Ctrl-J 新行	\012
\r	Ctrl-M 回车	\015
\t	Ctrl-I tab键	\011
\v	Ctrl-X	\030

tr中特定控制字符的不同表达方式

8.1.2 去除重复出现的字符

下面文件包含了一些打印错误。这种情况时常发生，例如在vi编辑器中，偶尔按住一个键不放。

```
$ pg oops.txt
And the cowwwwws went homeeeeeeeee
Or did theyyyy
```

如果要去除重复字母或将其压缩在一起，使用-s选项。因为都是字母，故使用[a-z]。输入文件重定向到tr命令。

```
$ tr -s "[a-z]"< oops.txt
And the cows went home
Or did they

$ cat oops.txt | tr -s "[a-z]"
And the cows went home
Or did they
```

如果使用cat命令，再将结果管道输出至tr，结果是一样的

8.1.3 删除空行

要删除空行，可将之剔出文件。下面是一个文件**plane.txt**。文本间有许多空行。

```
$ pg plane.txt
987932   Spitfire



190992   Lancaster


238991   Typhoon
```

使用-s来做这项工作。换行的八进制表示为\012，命令为：

```
$ tr -s "[\012]" < plane.txt
987932   Spitfire
190992   Lancaster
238991   Typhoon
```

也可以使用换行速记方式\n

```
$ tr -s ["\n"] < plane.txt
987932   Spitfire
190992   Lancaster
238991   Typhoon
```

8.1.4 大写到小写

除了删除控制字符，转换大小写是tr最常用的功能。为此需指定即将转换的小写字符[a - z]和转换结果[A - Z]。

第一个例子(输出有误)，tr从一个包含大小写字母的字符串中接受输入。

```
$ echo "May Day, May Day, Going Down.." | tr "[a-z]" "[A-Z]"
MAY DAY, MAY DAY, GOING DOWN..
```

8.1.5 小写到大写

转换小写到大写与上一节大写到小写过程刚好相反。

```
$ echo " Look for the route, or make the route" | tr "[a-z]" "[A-Z]"
 LOOK FOR THE ROUTE, OR MAKE THE ROUTE
```

8.1.6 删除指定字符

偶尔会从下载文件中删除只包含字母或数字的列。需要结合使用-c和-s选项完成此功能。

下面的文件包含一个星期的日程表。任务是从其中删除所有数字，只保留日期。日期有大写，也有小写格式。因此需指定两个字符范围[a-z]和[A-Z],

-s 选项表明压缩所有新行， -c 表明保留所有字母不动。

```
$ pg diary.txt
monday 10:50
Tuesday 15:30
wednesday 15:30
thurday 10:30
Friday 09.20
$ tr -cs "[a-z][A-Z]" "[\012*]" < diary.txt
monday
Tuesday
wednesday
thurday
Friday
```

8.1.7 转换控制字符

tr的第一个功能就是转换控制字符，特别是从dos向UNIX下载文件时，忘记设置ftp关于回车换行转换的选项时更是如此。

下面是故意没有设置转换开关的一个文本文件，是关于文具需求的一部分内容。使用cat -v显示控制字符。

```
$ cat -v stat.tr
Boxes paper^^^^^^12^M
Clips metal^^^^^^50^M
Pencils-meduim^^^^^^10^M
^Z
```

猜想‘^ ^ ^ ^ ^ ^’是tab键。每一行以Ctrl - M结尾，文件结尾Ctrl-Z，以下是改动方法。

使用-s选项，查看ASCII表。^的八进制代码是136，^M是015，tab键是011，^Z是032 ,下面将按步骤完成最终功能:

用tab键替换^^^^^^，命令为" \ 136 " " [\ 011 *] "。将结果重定向到临时工作文件stat.tmp。

```
$ tr -s "[\136]" "[\011*]" < stat.tr  >stat.tmp
Boxes paper 12^M
Clips metal 50^M
Pencils-medium 10^M
^Z
```

用新行替换每行末尾的^M，并用\n去除^Z，输入要来自于临时工作文件**stat.tmp**

```
$ tr -s "[\015\032]" "\n" < stat.tmp
Boxes paper      12
Clips metal      50
Pencils-medium   10
```

8.1.8 快速转换

替换**passwd**文件中所有冒号，代之以**tab**键，可以增加可读性。将冒号引起来，指定替换字符串中**tab**键八进制值**011**，下面是**passwd**文件，后跟**tr**命令结果:

```
$ pg passwd
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:

$ tr -s "[:]" "[\011]" < passwd
halt    *       7       0       halt    /sbin   /sbin/halt
mail    *       8       12      mail    /var/spool/mail
news    *       9       13      news    /var/spool/news
uucp    *       10      14      uucp    /var/spool/uucp
```

8.1.9 匹配多于一个字符

可以使用[character*n]格式匹配多于一个字符。下述文件列出系统硬盘信息，其中包含了系统已经注册的和未识别的。第一列是数字，如果不全是0，表明第二列相应硬盘已经注册。

有时全部为0看起来很烦人，找个吸引人注意力的符号来代替它，以便一眼就能看出哪个硬盘已注册，哪个不可识别。原文件如下：

```
$ pg hdisk.txt
1293 hdisk3
4512 hdisk12
0000 hdisk5
4993 hdisk12
2994 hdisk7
```

从文件列表中知道，有一个硬盘未注册，因此用星号代替所有的0。模式为[0 * 4]，意即匹配至少4个0，替换字符串为星号，过滤命令及结果如下：

```
$ tr "[0*4]" "*" <hdisk.txt
1293 hdisk3
4512 hdisk12
**** hdisk5
4993 hdisk12
2994 hdisk7
```