tr命令

tr 命令可以用来删除一段文本信息中的某些文字。或者将其进行转换。

使用方式：

```
tr [option]...SET1 [SET2]
```

常用的选项有：

选项	说明
-d	删除和set1匹配的字符，注意不是全词匹配也不是按字符顺序匹配
-s	去除set1指定的在输入文本中连续并重复的字符

操作举例：

```
# 删除 "hello shiyanlou" 中所有的'o','l','h'
$ echo 'hello shiyanlou' | tr -d 'olh'
# 将"hello" 中的ll,去重为一个l
$ echo 'hello' | tr -s 'l'
# 将输入文本，全部转换为大写或小写输出
$ echo 'input some text here' | tr '[:lower:]' '[:upper:]'
# 上面的'[:lower:]' '[:upper:]'你也可以简单的写作'[a-z]' '
```

col 命令

col 命令可以将Tab换成对等数量的空格键，或反转这个操作。

使用方式：

```
col [option]
```

常用的选项有：

选项	说明
-x	将Tab转换为空格
-h	将空格转换为Tab（默认选项）

操作举例：

```
# 查看 /etc/protocols 中的不可见字符，可以看到很多 ^I ，这其实就是 Tab 转义成可见字符的符号
$ cat -A /etc/protocols
# 使用 col -x 将 /etc/protocols 中的 Tab 转换为空格,然后再使用 cat 查看，你发现 ^I 不见了
$ cat /etc/protocols | col -x | cat -A
```

join命令

用于将两个文件中包含相同内容的那一行合并在一起。

使用方式：

```
join [option]... file1 file2
```

常用的选项有：

选项	说明
-t	指定分隔符，默认为空格
-i	忽略大小写的差异
-1	指明第一个文件要用哪个字段来对比，默认对比第一个字段
-2	指明第二个文件要用哪个字段来对比，默认对比第一个字段

操作举例：

```
$ cd /home/shiyanlou
# 创建两个文件
$ echo '1 hello' > file1
$ echo '1 shiyanlou' > file2
$ join file1 file2
# 将/etc/passwd与/etc/shadow两个文件合并，指定以':'作为分隔符
$ sudo join -t':' /etc/passwd /etc/shadow
# 将/etc/passwd与/etc/group两个文件合并，指定以':'作为分隔符，分别比对第4和第3个字段
$ sudo join -t':' -1 4 /etc/passwd -2 3 /etc/group
```

paste命令

paste这个命令与join命令类似，在不对比数据的情况下，简单地将多个文件合并一起，以Tab隔开。

使用方式：

```
paste [option] file...
```

常用的选项有：

选项	说明
-d	指定合并的分隔符，默认为Tab
-s	不合并到一行，每个文件为一行

操作举例：

```
$ echo hello > file1
$ echo shiyanlou > file2
$ echo www.shiyanlou.com > file3
$ paste -d ':' file1 file2 file3
$ paste -s file1 file2 file3
```

grep命令

grep命令用于打印输出文本中匹配的模式串，它使用正则表达式作为模式匹配的条件。grep支持三种正则表达式引擎，分别用三个参数指定：

参数	说明
-E	POSIX扩展正则表达式，ERE
-G	POSIX基本正则表达式，BRE
-P	Perl正则表达式，PCRE

不过在你没学过perl语言的大多数情况下你将只会使用到ERE和BRE,所以我们接下来的内容都不会讨论到PCRE中特有的一些正则表达式语法（它们之间大部分内容是存在交集的，所以你不用担心会遗漏多少重要内容）

在通过grep命令使用正则表达式之前，先介绍一下它的常用参数：

参数	说明
-b	将二进制文件作为文本来进行匹配
-c	统计以模式匹配的数目
-i	忽略大小写
-n	显示匹配文本所在行的行号
-v	反选，输出不匹配行的内容
-r	递归匹配查找
-A n	n为正整数，表示after的意思，除了列出匹配行之外，还列出后面的n行
-B n	n为正整数，表示before的意思，除了列出匹配行之外，还列出前面的n行
--color=auto	将输出中的匹配项设置为自动颜色显示

使用基本正则表达式，BRE

- 位置

查找/etc/group文件中以"shiyanlou"为开头的行

```
1 $ grep 'shiyanlou' /etc/group
2 $ grep '^shiyanlou' /etc/group
```

- 数量

```
# 将匹配以'z'开头以'o'结尾的所有字符串
$ echo 'zero\nzo\nzoo' | grep 'z.*o'
# 将匹配以'z'开头以'o'结尾，中间包含一个任意字符的字符串
$ echo 'zero\nzo\nzoo' | grep 'z.o'
# 将匹配以'z'开头,以任意多个'o'结尾的字符串
$ echo 'zero\nzo\nzoo' | grep 'zo*'
```

注意：其中\n为换行符

- 选择

```
# grep默认是区分大小写的，这里将匹配所有的小写字母
$ echo '1234\nabcd' | grep '[a-z]'
# 将匹配所有的数字
$ echo '1234\nabcd' | grep '[0-9]'
# 将匹配所有的数字
$ echo '1234\nabcd' | grep '[[:digit:]]'
# 将匹配所有的小写字母
$ echo '1234\nabcd' | grep '[[:lower:]]'
# 将匹配所有的大写字母
$ echo '1234\nabcd' | grep '[[:upper:]]'
# 将匹配所有的字母和数字，包括0-9,a-z,A-Z
$ echo '1234\nabcd' | grep '[[:alnum:]]'
# 将匹配所有的字母
$ echo '1234\nabcd' | grep '[[:alpha:]]'
```

下面包含完整的特殊符号及说明：

特殊符号	说明
[:alnum:]	代表英文大小写字母及数字，亦即 0-9, A-Z, a-z
[:alpha:]	代表任何英文大小写字母，亦即 A-Z, a-z
[:blank:]	代表空白键与 [Tab] 按键两者
[:cntrl:]	代表键盘上面的控制按键，亦即包括 CR, LF, Tab, Del.. 等等
[:digit:]	代表数字而已，亦即 0-9
[:graph:]	除了空白字节 (空白键与 [Tab] 按键) 外的其他所有按键
[:lower:]	代表小写字母，亦即 a-z

[:print:]	代表任何可以被列印出来的字符
[:punct:]	代表标点符号 (punctuation symbol)，亦即：" ' ? ! ; : # $...
[:upper:]	代表大写字母，亦即 A-Z
[:space:]	任何会产生空白的字符，包括空白键, [Tab], CR 等等
[:xdigit:]	代表 16 进位的数字类型，因此包括： 0-9, A-F, a-f 的数字与字节

注意：之所以要使用特殊符号，是因为上面的[a-z]不是在所有情况下都管用，这还与主机当前的语系有关，即设置在LANG环境变量的值，zh_CN.UTF-8的话[a-z]，即为所有小写字母，其它语系可能是大小写交替的如，"a A b B...z Z"，[a-z]中就可能包含大写字母。所以在使用[a-z]时请确保当前语系的影响，使用[:lower:]则不会有这个问题。

```
# 排除字符
$ $ echo 'geek\ngood' | grep '[^o]'
```

注意:当^放到中括号内为排除字符，否则表示行首。

使用扩展正则表达式，ERE

要通过grep使用扩展正则表达式需要加上-E参数，或使用egrep。

- 数量

```
# 只匹配"zo"
$ echo 'zero\nzo\nzoo' | grep -E 'zo{1}'
# 匹配以"zo"开头的所有单词
$ echo 'zero\nzo\nzoo' | grep -E 'zo{1,}'
```

注意：推荐掌握{n,m}即可，+,?,*，这几个不太直观，且容易弄混淆。

- 选择

```
# 匹配"www.shiyanlou.com"和"www.google.com"
$ echo 'www.shiyanlou.com\nwww.baidu.com\nwww.google.com' | grep -E 'www\.(shiyanlou|google)\.com'
# 或者匹配不包含"baidu"的内容
$ echo 'www.shiyanlou.com\nwww.baidu.com\nwww.google.com' | grep -Ev 'www\.baidu\.com'
```

注意：因为.号有特殊含义，所以需要转义。

seq

sed 命令基本格式：

```
sed [参数]... [执行命令] [输入文件]...
```

```
# 形如：
$ sed -i 's/sad/happy/' test # 表示将test文件中的"sad"替换为"happy"
```

参数	说明
-n	安静模式，只打印受影响的行，默认打印输入数据的全部内容
-e	用于在脚本中添加多个执行命令一次执行，在命令行中执行多个命令通常不需要加该参数
-f filename	指定执行filename文件中的命令
-r	使用扩展正则表达式，默认为标准正则表达式
-i	将直接修改输入文件内容，而不是打印到标准输出设备

sed执行命令格式：

```
[n1][,n2]command
[n1][~step]command
# 其中一些命令可以在后面加上作用范围，形如：
$ sed -i 's/sad/happy/g' test # g表示全局范围
$ sed -i 's/sad/happy/4' test # 4表示指定行中的第四个匹配字符串
```

其中n1,n2表示输入内容的行号，它们之间为,逗号则表示从n1到n2行，如果为~波浪号则表示从n1开始以step为步进的所有行；command为执行动作，下面为一些常用动作指令：

命令	说明
s	行内替换
c	整行替换
a	插入到指定行的后面
i	插入到指定行的前面
p	打印指定行，通常与-n参数配合使用
d	删除指定行

apt-get

apt-get 是用于处理 apt包的公用程序集，我们可以用它来在线安装、卸载和升级软件包等，下面列出一些apt-get包含的常用的一些工具：

工具	说明

install	其后加上软件包名，用于安装一个软件包
update	从软件源镜像服务器上下载/更新用于更新本地软件源的软件包列表
upgrade	升级本地可更新的全部软件包，但存在依赖问题时将不会升级，通常会在更新之前执行一次update
dist-upgrade	解决依赖关系并升级(存在一定危险性)
remove	移除已安装的软件包，包括与被移除软件包有依赖关系的软件包，但不包含软件包的配置文件
autoremove	移除之前被其他软件包依赖，但现在不再被使用的软件包
purge	与remove相同，但会完全移除软件包，包含其配置文件
clean	移除下载到本地的已经安装的软件包，默认保存在/var/cache/apt/archives/
autoclean	移除已安装的软件的旧版本软件包

下面是一些apt-get常用的参数：

参数	说明
-y	自动回应是否安装软件包的选项，在一些自动化安装脚本中使用这个参数将十分有用
-s	模拟安装
-q	静默安装方式，指定多个q或者-q=#,#表示数字，用于设定静默级别，这在你不想要在安装软件包时屏幕输出过多时很有用
-f	修复损坏的依赖关系
-d	只下载不安装
--reinstall	重新安装已经安装但可能存在问题的软件包
--install-suggests	同时安装APT给出的建议安装的软件包

```
# 更新软件源
$ sudo apt-get update
# 升级没有依赖问题的软件包
```

```
$ sudo apt-get upgrade
# 升级并解决依赖关系
$ sudo apt-get dist-upgrade
```

dpkg

`dpkg`常用参数介绍：

参数	说明
`-i`	安装指定deb包
`-R`	后面加上目录名，用于安装该目录下的所有deb安装包
`-r`	remove，移除某个已安装的软件包
`-I`	显示`deb`包文件的信息
`-s`	显示已安装软件的信息
`-S`	搜索已安装的软件包
`-L`	显示已安装软件包的目录信息