文本处理

1. 文本应用程序

   UNIX/Linux 严重依赖于文本文件进行某些数据类型的存储。如文件，网页，电子邮件，打印机输出，程序源代码等。

   例子：

   `cat > foo.txt` 输入一些文字，制表符，回车内容后按 Ctrl-D 告诉 cat 达到文件末尾，可以建立一个新文件。

   `cat -A foo.txt` 会看到制表符使用^I，$表示行末

   注意 UNIX 中的回车是 ASCII 编码 10 即换行符，Windows 中的回车是回车符（ASCII 码是 13）和换行符的组合。

   `cat -ns foo.txt` -n 参数表示在输出中加上行号，-s 如果有连续多个空行，则只输出一个。

   `sort > foo.txt` 建立一个排序后的文件

选项	描述
-b	默认情况下，整行都会排序。该选项忽略行开头的空格，并从第一个非空白字符开始排序
-f	排序的时候不区分大小写
-n	该参数将字符串作为数值进行大小比较，否则直接按照字典顺序
-r	逆序排序
-k *field1, filed2*	对 *field1* 与 *field2* 之间的字符排序，而不是整个文本行
-m	将每个输入参数当作已经排序的文件名。将多个文件合并为一个排序好的文件
-o	将排序结果输出到文件而不是标准输出
-t *separator-char*	将 *separator-char* 作为字段分隔符，默认情况下是制表符

   例子：

   ```
   du -s /usr/share/* | sort -nr | head
   ```

   取出文件夹大小最大的前 10 个(head)行，按逆序输出

   ```
   ls -l /usr/bin | sort -nr -k 5 | head
   ```

   查看/usr/bin 目录下，按大小排序，找出其中 5 个最大的，逆序输出

   ```
   sort -k 3.7nbr -k 3.1nbr -k 3.4nbr distros.txt
   ```

   对 distros.txt 文件排序，首先按照第 3 个字段的第 7 个字母降序排序，再按第三个字段的第一个字段降序排序，最后按照第 3 个字段的第 4 个字符进行降序排序

   ```
   sort -t ':' -k 7 /etc/passwd | head
   ```

   按照冒号分隔文件行，按第 7 个字段进行排序，取前 10 个

   uniq 命令用于移除重复行，通常和 sort 命令结合使用。但是 GUN 版本的 sort 命令也支持-u 选项，达到和 uniq 同样的效果。

选项	功能描述
`-c`	输出重复行列表，并且在重复行上加上重复次数
`-d`	只输出重复行，而不包含单独行
`-f` *n*	忽略每行前 *n* 个字段。字段分隔符只能是空格
`-i`	忽略大小写
`-s` *n*	跳过每行前 *n* 个字符
`-u`	仅输出不重复的行，该选项是默认的

2. 切片和切块

   - cut——删除文本行中的部分内容

     cut 命令用于从文本行中提取一段文字并将其输出至标准输出。可以接受多个文件和标准输入作为输入参数。

选项	含义

`-c char_list`	从文本行中提出 char_list 定义的内容
`-f field_list`	从文本行中提出 field_list 定义的一个或多个字段
`-d delim_char`	指定-f 选项后使用 delim_char 作为分隔符，默认分隔符是制表符
`--complement`	从文本中提出整行，除了那些-c 和/或-f 指定的部分

`cut -f 5 foo.txt` 取出 foo.txt 文件的第 5 个字段输出到控制台上

`cut -d ' ' -f 1 distros.txt` 取出 distros.txt 的第一个字段，以空格作为分隔字段的标准

- paste——合并文本行

  paste 是 cut 命令的逆操作。用于向文件中增加一个或者更多的文本。

  ```
  paste distros-dates.txt distros-versions.txt
  ```

- join——连接两个文本文件中具有相同字段的行

  join 和 paste 类似，都是进行文件信息的增加。但是类似于数据库中表格的关联。

  ```
  paste distros-dates.txt distros-versions.txt
  ```

3. 文本比较

   文本比较通常用于与其它文件进行内容比较，管理员经常利用该功能进行漏洞检查。

   - comm——逐行比较两个已排序文件

     输出的第一列是第一个文件中独有的，第二列是第二个文件独有的，第三列是共有的

     该命令还有 n 参数，可以取值 1，2，3.用于省略指定的列。

     例子： `comm -12 foo.txt` 用于省略输出的前两列，只查看相同部分

   - diff——逐行比较文件

     ```
     diff file1 file2
     ```

改变操作	功能描述
`r1ar2`	将第二个文件中 r2 位置的行添加到第一个文件的 r1 位置处
`r1cr2`	用第二个文件 r2 处的行替代第一个文件中 r1 位置的行
`r1dr2`	删除第一个文件中 r1 处的行，并且将删除内容放入 r2 行

上面的格式输出并不常见，更常用的是上下文格式和统一格式。

diff 上下文格式差异标识符

标识符	含义
（无）	该行表示两个文件共有的行
-	表示缺少的行
+	表示多余的行
!	表示改变的行

`diff -c file1 file2` -c 参数表示使用上下文格式进行输出

`diff -u file1 file2` -u 参数表示使用统一格式进行输出

diff 统一格式差异标识符

标识符	含义
（无）	两个文件共有的行
-	相对于第二个文件而言，第一个文件中没有的行
+	第一个文件中多余的行

- patch——对原文件进行 diff 操作

  patch 命令主要用于更新文本文件。语法格式为：

  `patch < diff_file` 其中 *diff_file* 是 diff 文件输出的结果，该文件用以下命令生成：

  ```
  diff -Naur old_file new_file > diff_file
  ```

patch 主要用于修补源代码中的问题。整个源代码树相对于 patch 文件来讲会大得多。而且 patch 命令的结果比较易读。

4. 非交互式文本编辑

- tr——替换或删除字符

```
echo "lowercase letters" | tr a-z A-Z
```

参数	含义
d	参数 d 指从标准输入中删除指定的字符
s	参数 s 指删除指定的重复字符串

```
echo "aaabbbccc" | tr -s ab
```

`tr -d '\r' dosFile unixFile` 将 DOS 格式文件转换为 Windows 格式文件

- sed——用于文本过滤和转换的流编辑器

sed 是 Stream Editor 的缩写。总的来说，sed 给定某个简单命令或者包含命令的脚本文件，然后 sed 对文本流中的内容执行给定的编辑命令。例如：

```
echo "front" | sed 's/front/back/'
```
将 front 替换为 back，s 是替换命令

一般来说 sed 命令总是以单个字母开头，后面根一个分割线。分割线是任意的，总是由命令后面的第一个字符决定，如

```
echo "front" | sed 's_front_back_'
```

sed 的多数命令允许在其前面添加一个地址，该地址用来指定输入流的哪一行被编辑。如果地址被省略，则默认对每一行执行该编辑命令。

```
echo "front" | sed '1s/front/back/'
```

地址

地址格式	说明
n	n 是正整数表示的行号
$	最后一行
/regexp/	使用 POSIX 基本正则表达式描述的行
addr1,addr2	行范围，从 addr1 到 addr2 的所有行
first~step	从第 first 行开始，每隔 step 行的所有行
addr1, +n	从 addr1 开始之后的 n 行
addr!	除了第 addr 行之外的所有行

```
sed -n '1,5p' distros.txt
```

选项 n 指不会自动打印，否则 sed 会输出每一行的内容。p 命令指只输出指定匹配行的内容。该命令输出 distros.txt 的前 5 行

```
sed -n '/SUSE/p' distros.txt      #输出包含 SUSE 的行
```

sed 的基本编辑指令

命令	功能描述
=	输出当前行号
a	在当前行后附加文本
d	删除当前行
i	在当前行前输入文本
p	打印当前行
q	退出 sed 不再执行
s/regexp/replacement/	替换
y/set1/set2	将字符集 set1 转化为字符集 set2。sed 要求这两个字符串等长

例子：

```
sed 's/\([0-9]\{2\}\)\/\([0-9]\{2\}\)\/\([0-9]\{4\}\)$/\3-\1-\2/' distors.txt
```

该命令对日期格式做了替换，首先查找 MM/DD/YYYY, 将其更换为 YYYY/MM/DD

- aspell——交互式拼写检查工具

  `aspell check textfile` #检查 *`textfile`* 中的语法错误

5. 简单的格式化工具
   - nl——对行进行标号

选项	含义
-b style	按照 style 格式对正文进行编号， a 是对每行编号 t 仅仅对非空白行编号，这是默认选项 n 不对任何行编号 pregexp 只对与基本正则表达式匹配的行编号
-f style	以 style 的格式对页脚进行编号，默认选项是 n（无）
-h style	以 style 的格式对页眉进行编号，默认选项是 n（无）
-i number	设置页编号的步长为 number。默认为 1
-n format	设置编号格式为 format，其中的 format 可以是： ln 左对齐，无缩进 rn 右对齐，无缩进。这是默认选项 rz 右对齐，右缩进
-p	在每个逻辑页的开始不再进行编码重置
-s string	在每行行号后面增加 string 作为分隔符。默认情况下是 tab 制表符
-v number	将每个逻辑页的第一个行号设置为 number，默认是 1
-w width	设置行号字段的宽度为 width，默认为 0

   - fold——将文本中行的长度限制为指定长度

     默认长度是 80 个字符

     ```
     echo "The quick brown fox jumped over lazy dog." | fold -w 12
     ```

   - fmt——简单的文本格式化工具

     fmt -w 50 fmt-info.txt | head

     默认情况下，空白行，单词之间的空格和缩进都保留在输出结果中；不同缩进量的连续输入行并不进行拼接；制表符会在输入中扩展并直接输出。

选项	功能描述
-c	在“冠边缘”模式下运行。此模式保留段落前两行的缩进，随后的行都与第二行的缩进对齐
-p *string*	只格式化 *string* 开头的行。
-s	仅截断行模式。之后根据列宽进行截断。短行不会与其他行进行合并。此模式适合代码。
-u	字符间隔统一
-w width	指定行宽不超过 width 个字符

   - pr——格式化打印文本

     该命令用于给文本标记页码，通常适合将内容分为几页，并且每页顶部和底部都留出空白行，插入页眉和页脚。

   - printf——格式化并打印数据