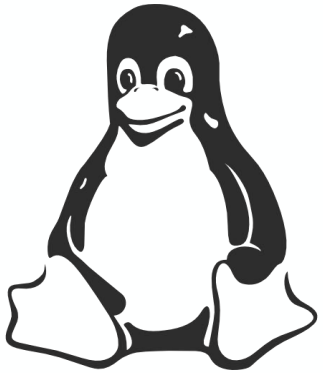


Linux系统管理

南通师范高等专科学校 朱亚林

本章导言

搜索是信息社会获取知识的重要方法。方法得当，事半功倍。搜索时，最理想的状态就是能够掌握目标对象的具体信息，然后交由搜索工具去查询，但并不是每次搜索都能这样具象的去表达，特别是有时要搜索的对象并不是一个，而是特征相似的一组时，就要采用一种比较“模糊”的范式来完成。

本章将详细讲述正则表达式及其应用，让你掌握如何更为科学和全面的通过表达式检索、过滤文件信息。

第7章

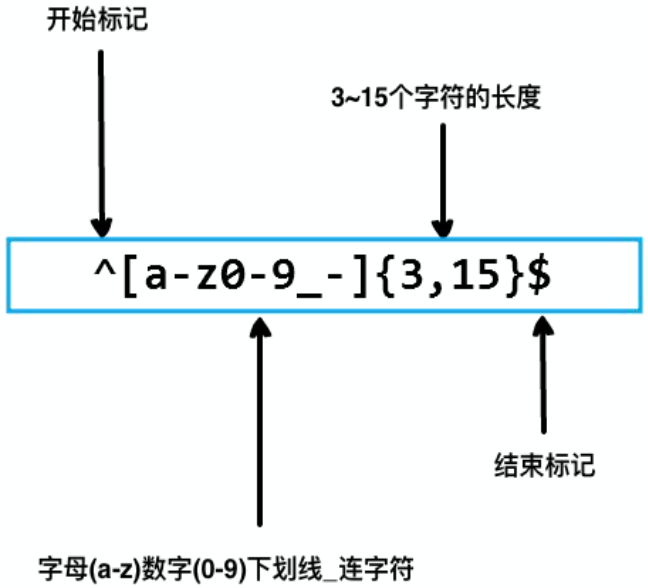
信息的处理：正则表达式与grep、sed、awk的应用

7.1.1 正则表达式的源起

- 上个世纪40年代，美国两位神经生理方面的科学家Warren McCulloch和Walter Pitts研究出了一种用数学方式来描述神经网络的新方法，他们创造性地将神经系统中的神经元描述成了小而简单的自动控制元。
- 1951年，一位名叫Stephen Kleene的数学科学家，他在Warren McCulloch和Walter Pitts早期工作的基础之上，发表了一篇题目是《神经网事件的表示法》的论文中引入了正则表达式的概念。
- 60年代，Unix 之父Ken Thompson发表了《正则表达式搜索算法》论文，将正则表达式应用于计算搜索算法的一些早期研究。之后，Ken Thompson又将正则表达式引入编辑器QED，以及Unix上的编辑器ed，并最终引入grep命令中。
- 70年代，Alfred Aho编写了egrep程序（其中e表示加强版的意思）对grep功能进行增强。随后awk、sed等工具也开始百花齐放。

7.1.2 正则表达式的定义

所谓正则表达式，是对字符串操作的一种逻辑公式，即用事先定义好的一些特定字符、及这些特定字符的组合，组成一个“规则字符串”，这个“规则字符串”用来表达对字符串的一种过滤逻辑。



7.1.3 Linux与正则表达式

- 正则表达式的应用范围很广，在它的发展成熟过程中也衍生出了多个标准。在Linux中主要遵循POSIX规范，Linux中将正则表达式分为两类。
- 一类叫做基本正则表达式（BRE，basic regular expression）
- 另一类叫做扩展正则表达式（ERE,extended regular expression）

7.1.3 Linux与正则表达式

对于Linux操作系统而言，主要注意以下两点：

- Linux正则表达式以行为单位处理字符。也就是说Linux中使用正则表达式过滤内容时，是一行一行的进行匹配的。
- 为了保证Linux中正则表达式的准确使用，需要在用户配置文件中设置LC_ALL=C，或是在使用前在命令行中输入export LC_ALL=C。

7.2 正则表达式的特殊字符

字符	含义
.	匹配单个任意字符
[0-9a-zA-Z]	匹配所有数字及大小写英文字母
[^字符]	匹配除[]内字符以外的所有字符
*	匹配前面的字符0次或多次
.*	表示0个或多个任意字符
+	匹配前一个字符 1 次或多次
{n}	表示重复前面的元素n次
{n,}	表示至少重复前面的元素n次
{m,n}	表示重复m到n次前面的元素
^	表示一行的开始位置
$	表示一行的结束位置
^$	表示一个空行
\	引用特殊字符
()	分组过滤，被括起来的元素表示一个整体，另外()的内容可以被后面的\n引用，n为数字，表示引用第几个括号的内容
\n	引用前面()小括号里的内容，如：(aa)\1，匹配aaaa

7.2 正则表达式的特殊字符

特殊符号	代表意义
[:alnum:]	代表英文大小写字符及数字，亦即 0-9， A-Z， a-z
[:alpha:]	代表任何英文大小写字符，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:]	除了空格字符（空白键和[Tab] 按键） 外的其他所有按键
[:lower:]	代表小写字符，亦即 a-z
[:print:]	代表任何可以被打印出来的字符
[:punct:]	代表标点符号 （punctuation symbol），亦即：" ' ? ! : # $...
[:upper:]	代表大写字符，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 的数字与字符

7.2 正则表达式的特殊字符

特殊字符与通配符的区别

- 这些特殊字符中有一部分与通配符较为接近，但是它们之间是有着本质的区别的。
- Linux系统中，正则表达式一般用来查询文件内容、文本和字符串，会与grep、egreap、sed、awk等命令配合使用
- 通配符一般用来查找文件名，常见的命令都支持，如ls、rm等。

温故知新

1.如何查询/bin目录下所有以“l”开头的文件?
2.如何查询/bin目录下，所有以“l”开头，仅含两个字母的文件?

7.2 正则表达式的特殊字符

结合Linux中正则表达式的两种类型基本正则表达式（BRE）和扩展正则表达式（ERE）来解读特殊字符，还需要注意以下两点：

- 基本正则表达式能够识别的元字符有“.”“^”“$”“*”“[]”等。
- 扩展正则表达式能够识别的元字符在基本正则表达式的基础上扩充了“?”“+”“()”、“|”“{}”。

7.3 grep命令

grep 命令用于查找文件里符合条件的字符串，如果发现被查找的文件中有符合查找条件的内容，默认情况下grep命令会把符合查询条件的那一行显示出来。

7.3 grep命令

- grep命令的一般格式如下：

```
grep [选项] 查找内容  文件名
```

7.3 grep命令

- grep命令的常用选项如下表所示

选项	作用
-e	实现多个选项间的逻辑or 关系
-E	开启扩展（Extend）的正则表达式
-i	忽略大小写（ignore case）
-v	反过来（invert），只打印没有匹配的，而匹配的反而不打印
-n	显示行号
-w	被匹配的文本只能是单词，而不能是单词中的某一部分
-c	显示被匹配到的行数，而不是显示被匹配到的内容
-o	只显示被模式匹配到的字符串
--color	将匹配到的内容以颜色高亮显示
-A n	显示匹配到的字符串所在的行及其后n行，after
-B n	显示匹配到的字符串所在的行及其前n行，before
-C n	显示匹配到的字符串所在的行及其前后各n行，context

7.3.1 简单搜索

grep命令的简单应用就是将待查询文件及查询字符串写入命令指定位置即可，以下以查询英汉字典为例。

案例：使用grep命令查英汉字典

（1）构建本例学习环境

http://t.edutest.fun/e

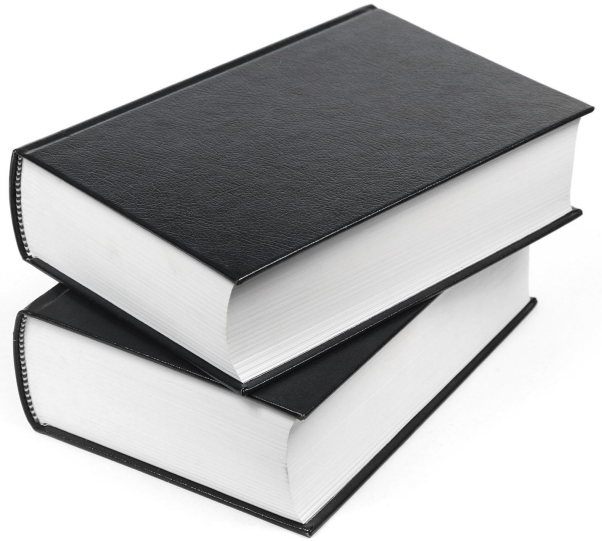
（2）使用grep命令查询包含“bag”字符串的单词

（3）使用grep命令查询包含“bag”字符串的行数

（4）使用grep命令从候选字符中进行查询　a?1e　[bcd]

（5）使用grep命令查询指定形式的单词

查找首字母是w，其后紧跟两个任意字符，并以字母d结束的单词



7.3.2 多文件搜索

grep的搜索操作不局限于某一个文件，可以是两个甚至多个，此时就要善用通配符来进行文件的匹配了。

案例：使用grep命令查多语言字典

（1）构建本例学习环境

http://t.edutest.fun/f

（2）查看dic2.txt中的文本内容

（3）查询中文“你好”，输出有关信息

（4）对查询信息进行精准匹配

（5）多个词典同时查询

（6）仅查询匹配信息和位置

7.3.3 扩展正则表达式

以上的应用主要是基本正则表达式的应用，如果有使用扩展正则表达式，就要加上grep命令的“-E”选项。

7.3.3 扩展正则表达式

案例：grep扩展正则表达式的使用

（1）指定字符重复出现 a opq 重复2 le

（2）指定内容重复出现 banana

7.3.4 正则表达式的再巩固

- 以下练习引用鸟哥的私房菜
- https://linux.vbird.org/linux_basic/centos7/0330regularex.php

7.3.4 正则表达式的再巩固

- 构建案例运行环境:
- wget -O regular_express.txt http://t.edutest.fun/m

例题一:搜索特定字符串

1. 搜索所有的“the”,并显示行号
2. 搜索所有没有“the”的行，显示行号
3. 不区分大小写，搜索所有的“the”，并显示行号

7.3.4 正则表达式的再巩固

- 构建案例运行环境:
- wget -O regular_express.txt http://t.edutest.fun/m

例题二：搜寻集合字符

1. 搜索 test 或 taste 这两个单词
2. 搜索所有含有 oo 的字符
3. 搜索含有oo的行，但不想要 oo 前面有 g
4. 搜索所有oo 前面不含有小写字符的行
5. 搜索所有包含数字的行

7.3.4 正则表达式的再巩固

- 构建案例运行环境：
- wget -O regular_express.txt http://t.edutest.fun/m

例题三：关于行首与行尾

1. 搜索以单词“the”开始的行
2. 搜索以小写字母开始的行
3. 搜索不是以字母开始的行
4. 搜索以符号“.”结束的行
5. 搜索文档中的空行
6. 显示/etc/rsyslog.conf中去除了空行及以“#”开头以后的内容

7.3.4 正则表达式的再巩固

- 构建案例运行环境：
- wget -O regular_express.txt http://t.edutest.fun/m

例题四：字符与字符的重复

1. 找出 g? ? d格式的字符串
2. 搜索至少两个 o 以上的字符串
3. 搜索字符串开头与结尾都是g，但是两个g之间仅能存在至少一个o

7.3.4 正则表达式的再巩固

- 构建案例运行环境:
- wget -O regular_express.txt http://t.edutest.fun/m

例题四：限定连续RE字符范围 {}

1. 搜索字符串开头与结尾都是g，但是两个g之间的o仅能重复2-5次

7.4 sed工具

- sed的全称是stream editor，它是一个非交互流编辑器，因此称之为工具。
- 使用sed处理文件的有着其独特的优势，它并不像vim一样将文件全部打开，再进入其中逐一编辑，而是以行为单位进行读取处理，处理完一行就立即打印出来，然后再进入下一行，直到全部处理结束。
- 使用sed处理文件无论文件大小，对内存消耗极低。

7.4 sed工具

- sed工具对于文件可以进行的操作包含添加、删除、插入、查找、替换等，一般只对读取的信息进行处理，不会影响到文件自身。其在命令行中一般的运行格式如下：

```
sed [选项] 命令 文件名
```

7.4 sed工具

- sed工具常用命令如下表所示

命令	作用
a	将命令之后的内容插入到匹配行之后一行
c	字符转换
d	删除行
i	将命令之后的内容插入到匹配行之前一行
p	打印匹配行内容
r	从外部文件中读入文本
s	使用正则表达式进行内容替换
w	追加写入文件
y	替换字符
=	打印当前行号
!	反向（对匹配条件以外的对象）进行操作

7.4 sed工具

- sed工具常用选项如下表所示

选项	作用
-e	多条件编辑
-f	指定sed脚本
-i	直接修改原文件
-n	仅显示处理后的结果信息

7.4.1 行的插入

- 文本的插入主要使用到两个命令，a或者i，a是在匹配条件后面一行插入新的信息，而i则是在匹配条件之前一行
- **案例：使用sed插入行**

1. 构建练习环境

```
wget -O MakeACircle.txt http://t.edutest.fun/g
```

2. 尝试在第一行之后插入指定信息“Hi Hi Hi”
3. 再次使用cat命令查看文件
4. 尝试在第一行之前插入指定信息“Hi Hi Hi”
5. 显示2-6行的内容

7.4.2 行的删除

- 行的删除涉及到的命令是d。使用d时，需要指定删除范围。
- **案例：使用sed删除行**

1. 删除第1行
2. 删除第5-8行
3. 删除第5行至结束
4. 删除包含指定文本“big”的行
5. 删除文件中所有的空行

7.4.3 字符的替换

- 使用sed的y命令可以对文本中的指定字符进行批量替换。只需要将需要被替换和替换的字符一一列出即可。
- **案例：使用sed替换字符**

1. 将目标文件中的有的小写字母s替换成了大写字母S
2. 将目标文件中所有的bdhsu替换成大写字母

7.4.4 字符串的查找与替换

- 使用sed工具进行字符串的查找与替换，其操作与字符的替换相似，只需要将y命令换成s即可。当然，在查找字符串时可以使用正则表达式来进行匹配。
- **案例：字符串的查找与替换**

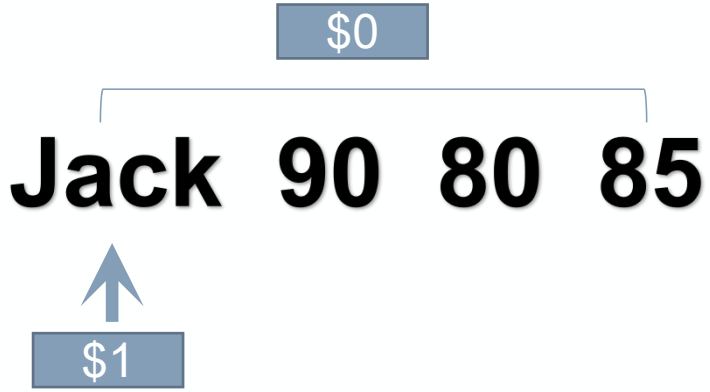
1. 查找所有的“hello”，将所有的“hello”替换成“Hi”
2. 将每行中第2次出现的“hello”替换成“HI”
3. 将第1行至第5行中所有的“hello”替换成“HI”
4. 将第5行中所有的“hello”、“Hello”都替换成了“HI”
5. 将所有的hello替换成HI,将所有的circle替换成star

7.4.5 操作的回存

- sed工具是对读出的信息进行操作，一般不会影响到文件自身。但如果需要将操作的结果进行保存，可以通过输出重定向或者加上“-i”来实现。

7.5 awk工具

- awk是一种处理文本文件的语言，是一个强大的文本分析工具。相对于grep的优势在于查找，sed的优势在于编辑，**awk的优势在于其对数据的分析**。
- awk从文件逐行读入信息，默认以空格为分隔符（也可以指定分隔符）将每行分段切片，然后分别对切开的部分进行分析处理。这些由分隔符分开的有效字段称之为“域”，自行首开始，从1进行递增计数，表示为$1。如果要表示全部域，则使用$0。



7.5 awk工具

- awk工具的一般使用格式如下：

```
awk [选项] 'BEGIN{ } / / {comand1;comand2} END{ }' 文件名
```

- 引号部分是指引用代码块，awk执行语句必须包含在内；
- BEGIN{ } 是指初始化代码块，在对每一行进行处理之前，初始化代码；
- / / 用来定义需要匹配的模式（字符串或者正则表达式），对满足匹配模式的行进行接下来的代码块操作；
- { } 命令代码块，包含一条或多条命令；
- END{ } 结尾代码块，在对每一行进行处理之后再执行的代码块，主要是进行最终计算或输出结尾摘要信息。

7.5 awk工具

- awk的常用选项如下表所示

选项	作用
-F	定义列分隔符
-f	指定调用脚本
-v	定义变量

7.5 awk工具

- awk中还有一些特殊标识，其作用如下表所示：

标识	作用
$0	表示整个当前行
$1	每行第一个字段
NF	字段数量变量
NR	每行的记录号，多文件记录递增
FNR	与NR类似，不过多文件记录不递增，每个文件都从1开始
FS	BEGIN时定义分隔符
RS	输入的记录分隔符，默认为换行符(即文本是按一行一行输入)
OFS	输出字段分隔符， 默认也是空格，可以改为制表符等

7.5.1使用awk显示信息

- awk工具显示信息的方式较为灵活，主要是使用print配合域来进行。
- **案例：基础信息的显示**

1. 构建本例实验环境

```
wget -O score.txt http://t.edutest.fun/h
```

2. 使用awk读取全部信息
3. 使用awk读取第1列和第3列信息
4. 替换score.txt中所有的制表符“/t”为“,”
5. 指定分隔符显示
6. 获取数据形状

7.5.2使用awk筛选信息

awk ‘{if(条件) 语句}’文件名

- 在进行信息显示时，还可以使用if语句对指定域加入条件，从而实现筛选功能。
- **案例：筛选信息**

1. 筛选art课程分数大于92的学生姓名
2. 筛选art课程分数大于92且Science课程分数大于95的学生姓名
3. 筛选art课程分数大于92或者Science课程分数大于95的学生姓名

7.5.3使用awk简单编程

- awk是一种文本文件处理语言，因此也可以进行程序的编写，同样也支持使用正则表达式进行条件的过滤。
- 在编写程序时，将需要初始化的全局变量放入BEGIN中，将最终总结信息放入END中，其余中间运算放入中间的{}中。
- **案例：筛选并计算**

1. 过滤Jack的信息(尝试使用正则表达式，以Jack开头)，并显示总分
2. 计算所有人的总分
3. 保存awk执行结果