Linux 操作系统

正则表达式

正则表达式

- 当从一个文件或命令输出中抽取或过滤文本时，可以使用正则表达式(rexp, regular expressions)
- 正则表达式是由普通字符和特殊字符的集合
- 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，并且还包括一些扩展的元字符集
- 正则表达式广泛使用在 grep、sed 命令和 awk 语言中

基本元字符集及其含义

^	只匹配行首（可以看成是行首的标志）
$	只匹配行尾（可以看成是行尾的标志）
*	一个单字符后紧跟 *，匹配 0 个或多个此单字符
[ ]	匹配 [] 内的任意一个字符 （[^] 反向匹配）
\	用来屏蔽一个元字符的特殊含义
.	匹配任意单个字符
c\{n\}	匹配 字符 c 连续出现 n 次的情形
c\{n,\}	匹配 字符 c 至少连续出现 n 次的情形
c\{n,m\}	匹配 字符 c 连续出现次数在 n 与 m 之间的情形

注：字符 c 可以通过 [ ]，\ 或 . 来指定，但只能是单个字符。
如：[a-z]\{5\}，\$\{2,\}，.\{2,5\}

元字符集举例

- 使用句点“ . ”匹配单字符
  - 匹配任意单个ASCII 字符，可以为字母或数字

```
..XC..      可以匹配 deXC1t、23XCdf
.w..w..w.   可以匹配 rwxrw-rw-
```

- 在行首以“ ^ ”匹配字符串或字符序列
  - 在一行的开始匹配字符或单词

```
^d      可以匹配 drwxr-xr-x、drw-r--r—
^.01    可以匹配 0011cx4、c01sdf
```

元字符集举例

- 在行尾以“ $ ”匹配字符串
  - 在行尾匹配字符串或字符，$ 符号放在匹配单词后面

trouble$	匹配以单词 **trouble** 结尾的所有行
^$	匹配所有空行

- 使用“ * ”匹配单个字符或其重复序列
  - 一个单字符后紧跟 *，表示匹配 **0 个或多个此字符**

compu*t	可以匹配 **comput、compuuut**
1013*	可以匹配 **1013、101333、101**

  - 注：星号必须跟其前面的字符结合才有意义

元字符集举例

- 使用“ \ ”屏蔽一个特殊字符的含义
  - 用来屏蔽一个元字符的特殊含义

```
\*\.pas$      匹配以 *.pas 结尾的所有行
```

- 使用“ [ ] ” 匹配一个字符范围或集合
  - 匹配“ [ ] ”内的字符，可以是单个字符，或字符序列，可以使用 - 表示一个字符序列范围，如 [A-Za-z0-9]
  - 当 [后面紧跟 ^ 符号时，表示不匹配方括号里内容

```
[Cc]omputer  匹配 Computer 和 computer
[^a-zA-Z]    匹配任一个非字母型字符
```

元字符集举例

- 使用“ \{ \} ”匹配模式出现的次数
  - c\{n\}：匹配 字符 c 连续出现 n 次的情形
  - c\{n,\}：匹配 字符 c 至少连续出现 n 次的情形
  - c\{n,m\}：匹配 字符 c 连续出现次数在 n 与 m 之间

A\{2\}B	只能匹配 AAB
A\{2,\}B	可以匹配 AAB 或 AAAAAB，但不能匹配 AB
A\{2,4\}B	匹配 AAB、AAAB、AAAAB 但不能匹配 AB 或 AAAAAB 等

- 实际上真正的格式是 {n}、{n,}、{n,m}，只不过对“{”和“}”用了 Esacpe 字符“\”

常用的正则表达式举例

[Ss]igna[lL]	匹配 signal、signaL、Signal、SignaL
[Ss]igna[lL]\.	同上，但后面加一句点
^USER$	只包含 USER 的行
\.	带句点的行
^d..x..x..x	用户、同组用户及其他用户都有可执行权限的目录
^[^s]	不以 s 开始的行
[yYnN]	大写或小写的 y 或 n
.*	匹配任意多个字符
^.*$	匹配任意行
^......$	只包含 6 个字符的行

常用的正则表达式举例

[a-zA-Z]	任意单个字母
[^a-zA-Z0-9]	非字母或数字
[^0-9\$]	非数字或美元符号
[123]	1 到 3 中一个数字
\^q	包含 ^q 的行
^.$	仅有一个字符的行
^\.[0-9][0-9]	以一个句点和两个数字开始的行

[0-9]\{2\}-[0-9]\{2\}-[0-9]\{4\} 日期格式 dd-mm-yyyy
[0-9]\{3\}\.[0-9]\{3\}\.[0-9]\{3\}\.[0-9]\{3\} 类 IP 地址格式 nnn.nnn.nnn.nnn

在 **shell** 编程中，一段好的脚本与完美的脚本之间的差别之一，就是要熟知正则表达式并学会使用它们。有很多可以处理文本的程序，比如 **grep**、**awk**、**sed** 等都使用正则表达式。