1. 正则表达式

参考book: The Linux Command Line

这里主要是指POSIX Basic Vs. Extended Regular Expressions，不是指perl用的那一套正则表达式。

POSIX basic 正则表达式一般叫做BRE， -- grep使用的。

POSIX Basic Vs. Extended Regular Expressions叫做ERE， -- egrep使用的, grep -E也是一样。

1.1 BRE

ERE和BRE都支持以下符号：

```
^ $ . [ ] *
```

上面这几个符号默认当特殊功能符号，如果需要匹配这些符号，需要在前面加\转义。

1.1.1 正常匹配

```
$ echo "haha [i889" | grep "\[i"    # 使用\[来匹配[符号
haha [i889
$ echo "haha [i889^" | grep  "\^"   # 匹配^字符
haha [i889^



$ echo "haha [i889" | grep "\[i8*"    # 匹配到两个8
haha [i889


$ echo "hahai" | grep "\(ha\)i"    # 使用括号的时候需要在前面加一个\,
hahai

$ echo "hahai" | grep "\(ha\)\+i"    # 使用+的时候也要在前面加一个\
hahai

$ echo "(hahai" | grep "(ha"
(hahai


##   ERE和BRE不支持\d类似表达数字的用法，一般建议直接使用[0-9]的方式实现匹配数字，用[a-zA-Z]匹配字母

# 匹配数字
$ echo "hahai89" | grep "hai[0-9]"
hahai89
```

```
$ echo "hahai89" | grep "hai[0-9]\+"
hahai89
$ echo "haha i889" | grep "8*"
haha i889




# 匹配字符
$ echo "haha i89" | grep "[a-zA-Z]\+ i89"
haha i89


# 匹配边界，使用\<和\>
$ echo "haha i89" | grep "\<i89\>"
haha i89
```

1.1.2 匹配非

```
$ echo "haha [i8899" | grep "[^89]"    # 匹配除8和9之外的所有字符
haha [i8899
```

像这种情况下，匹配到的是haha [i□ 当前行还是被匹配到的。如果不想匹配带89的行，需要在**grep**的参数加上-v选项。

1.1.3 grep选中不匹配的行

grep -v选项的解释

```
       -v, --invert-match
              Invert the sense of matching, to select non-matching lines.
(-v is specified by POSIX.)
```

```
$ echo "haha [i8899" | grep -v "89"    #  echo中带有89，grep使用了-v参数，所以该行不会被匹配到。
```

1.2 ERE增强

ERE支持BRE的所有规则，另外ERE比BRE多添加了对以下符号的支持：

```
( ) { } ? + |
```

注意:ERE多的这几个符号默认是当做特殊功能字符，如要其要被当前普通字符被匹配，需要在其前面加\进行转义。

```
$ echo "haha [i889" | grep -E "(88)"    # 匹配88, ()只起限定范围的使用
haha [i889
```

```
$ echo "haha [i889" | grep -E "89\>"    #匹配边界
haha [i889
$ echo "haha [i889" | grep -E "8+"     #匹配到两个8
haha [i889

$ echo "haha [i8899" | grep -E "88|99"    #匹配88或者99
haha [i8899


$ echo "haha) [i889" | grep -E "haha\)"    #匹配)符号
haha) [i889
```

```
{n} Match the preceding element if it occurs exactly n times.
{n,m} Match the preceding element if it occurs at least n times, but no
more than m times.
{n,} Match the preceding element if it occurs n or more times.
{,m} Match the preceding element if it occurs no more than m times.
```

1.3 sed命令

```
sed 's/regexp/replacement/' distros.txt

# regexp expand方式的匹配, 可以理解为ERE
sed -r 's/(regexp)/new_\1' distros.txt

#和上面效果一样, 是普通的regexp, 可以理解为BRE
sed 's/\(regexp\)/new_\1/' distros.txt


# -i 将替换后的结果写回原文件中
sed -i 's/\(regexp\)/new_\1/' distros.txt
```