2009-10-10凌晨5点左右，天气好像要晴.

第一章

行的起始和结束：脱字符『^』和美元符『$』

^cat 匹配以c作为一行的第一个字符，紧接着一个a，紧接着一个t的文本；

^cat$ 只包含cat的一行；

^$ 空行，没有任何字符，包括空白字符.

^ 无意义!因为每一行都有开头，所以每一行都能匹配——空行也不例外.

字符组(character class)：列出使用者在某处期望匹配的字符.

注意：在字符组以外，普通字符都有『接下来是』的意思，这与字符组内部的情况完全相反.字符组内部的内容是在『同一个位置』能够匹配的若干字符，它的意思是『或』.

只有在字符组内部，连字符如果没有出现在字符组开头(若出现在字符组开头，如[^-]或[-]，它就是一个普通字符)，它表示的就是一个元字符，表示一个范围.

若脱字符(^)出现在字符数组中，它就表示『排除』，即排除那些不希望出现的字符.

点号(dot)：匹配任意字符的『字符组』的简便写法.

但如果点号出现在字符组内部，那么它就不是一个元字符了.

多选结构：『|』

多选结构若出现在字符组中则不是一个元字符，而是一个普通字符.

字符组与多选结构的区别：一个字符组只能匹配目标文本中的单个字符，而每个多选结构自身都可能是完整的表达式，都可以匹配任意长度的文本.

多选结构通常都与()组合使用，()限定了多选的范围.

注意一下两个表达式的区别：

^From|Subject|Date: •

^(From|Subject|Date): •

忽略大小写：

正则中并没有忽略大小写的直观功能，但egrep中通过参数选项i来达到大小写忽略功能.

单词分界符：『\<』和『\>』

我们可以将它们理解为单词版本的^和$，用来匹配单词的开头和结束位置.

注意，’<’和’>’并不是元字符，只有跟’\’结合时，整个序列才具有特殊意义.

小结：

在字符组内部，元字符的定义规则以及其意义是不一样的，点号在字符组内部和外部的意义就不一样，相反连字符只有在字符组内部(但不能是开头位置)才是元字符(这是普遍情况)，否则就不是.

字符组和多选项不要混淆.无论列出有多少字符，字符组只能匹配一个字符，相反，多选项可以匹配任意长度的文本，每个多选项可能匹配的文本都是独立的，不过多选项没有像字符组那样的排除功能(字符组通过^来实现排除功能).

排除型字符组是表示所有未列出字符的简便方法.因此^x的意思并不是『只有当这个位置不是x时才能匹配』，而是说『匹配一个不等于x的字符』.其中的差别很细微，但很重要，例如前者的概念可以匹配一个空行，而^x则不行，这里在数量上只能匹配一个字符.

可选项元素：『？』

将『可选项』加在一个元素后面，就表示此处容许出现该元素，但它是否出现并非匹配成功的必要条件. 它只作用于之前紧邻的元素.

例子：(July|Jul) • (fourth|4th|4)使用’?’后可以简化为July? • (foourth|4(th)?)

重复出现：『+』和『\*』

与…?一样，…\*永远不会匹配失败的，区别只在于它们的匹配结果.

结合使用的复杂示例：<HR(•+SIZE•\*=•\*[0-9]+)? •\*>

重复区间：『...{min,max}』

括号及反向引用：

之前括号的作用是限制多选项的范围和将若干字符组合为一个单元.

反向引用容许我们匹配与表达式先前部分匹配的同样的文本.

括号能够”记忆”其中的子表达式匹配的文本，不能这些文本是什么，元字符序列\1都能记住它们.

一个可以穷举所有可能出现的重复单词的表达式：\<([A-Za-z]+) •+\1\>