正则表达式

regcomp

`int regcomp (regex_t *compiled, const char *pattern, int cflags)`
将我们写的正则表达式编译成计算机能识别的正则表达式

参数	含义
regex_t *compiled	存放编译后的正则表达式
const char *pattern	编译前的正则，就是我们写的正则
int cflags	一些编译选项

regex_t *compile

一个结构体数据类型
regex_t 的成员re_nsub 用来存储正则表达式中的子正则表达式的个数，子正则表达式就是用圆括号包起来的部分表达式

int cflags

flags 有如下4个值或者是它们或运算(|)后的值：

1. REG_EXTENDED 以功能更加强大的扩展正则表达式的方式进行匹配。
2. REG_ICASE 匹配字母时忽略大小写。
3. REG_NOSUB 不用存储匹配后的结果。
4. REG_NEWLINE 识别换行符，这样'$'就可以从行尾开始匹配，'^'就可以从行的开头开始匹配

例子

```
#include <regex.h>
regex_t reg;    //存放编译后的正则表达式
const char* pattern = "^\\w+([-+.]\\w+)*@\\w+([-.]\\w+)*.\\w+([-.]\\w+)*$";
regcomp(®, pattern, REG_EXTENDED);
```

regexec

`int regexec (regex_t *compiled, char *string, size_t nmatch, regmatch_t matchptr [], int eflags)`
执行正则表达式匹配字符串

参数	含义
regex_t *compiled	regcomp编译后的正则表达式
char *string	待匹配串
size_t nmatch	regmatch_t结构体数组的长度,即匹配结果数
regmatch_t matchptr []	存放匹配结果
int eflags	

regmatch_t

匹配到的结果, 形式不是直接是字符串, 而是下标

```
typedef struct
{
    regoff_t rm_so; //匹配到的字符串的起始下标，包含
    regoff_t rm_eo; //匹配到的字符串的结束下标，不包含
} regmatch_t;
```

可以看到传入的不是regmatch_t而是regmatch_t[]
因为匹配的不止一个结果, 还有子表达式, 就是括号
regmatch_t[0]是主正则表达式, regmatch_t[1]等是各个子正则表达式匹配的结果
主+子 总共有几个size_t nmatch指定了

int eflags

1. REG_NOTBOL 按我的理解是如果指定了这个值，那么'^'就不会从我们的目标串开始匹配。总之我到现在还不是很明白这个参数的意义
2. REG_NOTEOL 和上边那个作用差不多，不过这个指定结束end of line

例子

```
char* buf = "david19842003@gmail.com";
const size_t nmatch = 1;
regmatch_t pmatch[1];
int status = regexec(®, buf, nmatch, pmatch, 0);

status == REG_NOMATCH   //没匹配
status == 0 //匹配上了
```

regfree

`void regfree (regex_t *compiled)`

1. 清空compiled指向的regex_t结构体的内容
2. 如果是重新编译的话，一定要先清空regex_t结构体c

参考资料

[c regex 用法]