复习

2017年6月11日 8:49

1. xml
   - 特点:
     - 有一个文件头

       <?xml version="1.0" encoding="ut-8"?>
     - 由一系列的标签组成:逻辑上是树结构
       - 根标签只有一个
       - 一般成对出现<hello></hello>
       - 不成对:<hello/>
     - 标签区分大小写
     - 标签可以设置属性

     - 注释:

       <!-- xxxx -->
   - 中国
     - xx
     - xxx
     - xx
   - ./a.out
     - ldd a.out
2. Json
   - json数组
     - [字符串,布尔,整形,json数组,json对象]
   - json对象

     {

     key:value,

     key1:value

     }

键值:不能重复，必须是字符串

value:字符串,布尔,整形,json数组,json对象

- cjson

1Day

html简介

http://www.w3school.com.cn/html/index.asp
http://www.runoob.com/

1. HTML简介
   - HTML，Hyper Texture Markup Language，超文本标记语言。
   - 在计算机中以.html、.htm作为扩展名.
   - 可以被浏览器访问,就是经常见到的网页.
2. HTML特点
   - 语法非常简洁、比较松散，以相应的英语单词关键字进行组合
   - html标签不区分大小写
   - 大多数标签是成对出现的,有开始,有结束.
     - <html></html>
   - 不成对出现的称之为短标签
     -
 <hr/>
3. 标签中的属性和属性值
   - 属性="属性值"
     - hello, world
     - 属性值建议加引号,(双,单引号,不加都可以)
4. html组成部分
   - <!doctype html> 声明文档类型
   - <html>文档的头部好和主体内容 </html> 根标记
   - <head> 文档的头部信息</head> 头部标记 只能有一对
   - <title>显示在浏览器窗口的标题栏中“网页名称”</title> 位于<head>标记之内
   - <body></body> 主体标记位于<html>之内,<head>标记之后
   - 例：

```
<html>
    <head>
        <title>这是一个标题</title>
    </head>

    <body>
        <font color="red" size="5">hello, world</font>
```

```
    </body>
</html>
```

5. 注释:
    - <!-- 我是一个html注释 -->

文字和标题标签

1. 标题标签:
    - <h1></h1> // 最大
        - 只有一个
        - 搜索引擎优化: seo
    - <h2></h2>
    - ...
    - <h6></h6> // 最小
    - 1-6依次变小, 自动换行
2. 文本标签
    -
        - 属性:
            - color: 文字颜色
                - 表示方式:
                    - 英文单词: red green blue.....
                    - 使用16进制的形式表示颜色: #ffffff -- (rgb)
                    - 使用rgb(255, 255, 0)
            - size: 文字大小
                - 范围 1 -- 7
                    - 7最大
                    - 1最小
3. 文本格式化标签
    - 文本加粗标签
        -
        -
        - 工作里尽量使用strong
    - 文本倾斜标签
        -
        -
        - 工作里尽量使用em

- 删除线标签
  - <del></del>
  - <s></s>
  - 工作里尽量使用del
- 下划线标签（插入文本）
  - <ins></ins>
  - <u></u>
  - 工作里尽量ins

4. 段落：
   - xxx
     - 特点：
       - 上下自动生成空白行
5. 块容器：
   - This is a div element.
   - 用于没有语义含义的内容的块级容器(或网页的"划分")。
   - 属性：对齐方式
     - align
       - left
       - center
       - right
6. 换行
   -
7. 水平线
   - <hr/>
     - 属性：
       - color: 3种表示方法
       - size: 1-7
       - <hr color="red" size="3"/>

列表标签

1. 无序列表
    - 标签

           列表项

    - 属性: type
        - 实心圆圈: disc -- 默认
        - 空心圆圈: circle
        - 小方块: square
2. 有序列表
    - 标签

           列表项

    - 属性:
        - type -- 序号
            - 1 -- 默认
            - a
            - A
            - i -- 罗马数字(小)
            - I -- 罗马数字(大)
        - start
            - 从序号的什么位置开始表示
3. 自定义列表
    - 标签

```
<dl>
    <dt></dt> 小标题
        <dd></dd> 解释标题
        <dd></dd> 解释标题
</dl>
```

图片标签

图片标签

-
  - 属性:
    - src: 图片的来源 必写属性
    - alt: 替换文本 图片不显示的时候显示的文字
    - title: 提示文本 鼠标放到图片上显示的文字
    - width: 图片宽度
    - height: 图片高度
  - 注意:
    - 图片没有定义宽高的时候，图片按照百分之百比例显示
    - 如果只更改图片的宽度或者高度，图片等比例缩放。

超链接标签

1. 超链接标签
   - 超链接
     - 属性:
       - href: 去往的路径（跳转的页面）必写属性
       - title: 提示文本,鼠标放到链接上显示的文字
       - target
         - _self: 默认值 在自身页面打开（关闭自身页面，打开链接页面）
         - _blank:打开新页面 （自身页面不关闭，打开一个新的链接页面）
     - 示例:
       - 百度一下
2. 锚链接
   - 先定义一个锚点:<p id="sd">
   - 超链接到锚点:回到顶点

表格标签

1. 表格标签
   - <table></table>
     - 属性:
       - border -- 表格线, 宽度1-7
       - bordercolor -- 表格线颜色
       - width
       - height
   - <tr> -- 行
     - 属性
       - align -- 对齐方式
         - center
         - left
         - right
   - <td> -- 单元格(列)
     - 对其属性设置同tr
   - 示例:

```
<table border=>
    <tr>
        <td></td> 第一列
        <td></td> 第二列
    </tr>
    <tr>
        <td></td>
        <td></td>
    </tr>
    <tr>
        <td></td>
        <td></td>
```

```
    </tr>
</table>
```

http协议 - 应用层

- 请求消息(Request) - 浏览器给服务器发
  - 四部分: 请求行, 请求头, 空行, 请求数据
    - 请求行: 说明请求类型, 要访问的资源, 以及使用的http版本
    - 请求头: 说明服务器要使用的附加信息
    - 空行: 空行是必须要有的, 即使没有请求数据
    - 请求数据: 也叫主体, 可以添加任意的其他数据
  - 例:

```
GET /3.txt HTTP/1.1 \r\n
```

/: 资源目录的根目录

三部分内容由空格间隔

```
Host: localhost:2222
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:24.0) Gecko/201001  01 Firefox/24.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-cn,zh;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
If-Modified-Since: Fri, 18 Jul 2014 08:36:36 GMT
```

请求头: 由键值对构成的

请求数据

换行: \r\n

- 响应消息(Response) - 服务器给浏览器发
  - 四部分: 状态行, 消息报头, 空行, 响应正文
    - 状态行: 包括http协议版本号, 状态码, 状态信息
    - 消息报头: 说明客户端要使用的一些附加信息
    - 空行: 空行是必须要有的
    - 响应正文: 服务器返回给客户端的文本信息
  - 例:

```
HTTP/1.1 200 Ok
Server: micro_httpd
Date: Fri, 18 Jul 2014 14:34:26 GMT
Content-Type: text/plain; charset=iso-8859-1 (必选项)
```

告诉浏览器发送的数据是什么类型

```
Content-Length: 32
```

发送的数据的长度

```
Content-Language: zh-CN
Last-Modified: Fri, 18 Jul 2014 08:36:36 GMT
Connection: close

#include <stdio.h>
int main(void)
{
    printf("hello world!\n");
    return 0;
}
```

www.baidu.com

xx.tar.gz

x.tar.bz2

- HTTP1.1的五种请求方法

- GET
  - 请求指定的页面信息，并返回实体主体。
- POST
  - 向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。数据被包含在请求体中。POST请求可能会导致新的资源的建立和/或已有资源的修改。
- HEAD
  - 类似于get请求，只不过返回的响应中没有具体的内容，用于获取报头
- PUT
  - 从客户端向服务器传送的数据取代指定的文档的内容。
- DELETE
  - 请求服务器删除指定的页面。
- CONNECT
  - HTTP/1.1协议中预留给能够将连接改为管道方式的代理服务器。
- OPTIONS
  - 允许客户端查看服务器的性能。
- TRACE
  - 回显服务器收到的请求，主要用于测试或诊断。

- HTTP常用状态码

状态代码有三位数字组成，第一个数字定义了响应的类别，共分五种类别：

- 1xx：指示信息--表示请求已接收，继续处理
- 2xx：成功--表示请求已被成功接收、理解、接受
- 3xx：重定向--要完成请求必须进行更进一步的操作
- 4xx：客户端错误--请求有语法错误或请求无法实现
- 5xx：服务器端错误--服务器未能实现合法的请求

- 常见状态码：
  - 200 OK　　客户端请求成功
  - 400 Bad Request　　客户端请求有语法错误，不能被服务器所理解
  - 401 Unauthorized　　请求未经授权，这个状态代码必须和WWW-Authenticate报头域一起使用
  - 403 Forbidden　　服务器收到请求，但是拒绝提供服务
  - 404 Not Found　　请求资源不存在，eg：输入了错误的URL
  - 500 Internal Server Error　　服务器发生不可预期的错误
  - 503 Server Unavailable　　服务器当前不能处理客户端的请求，一段时间后可能恢复正常

http使用get和post请求数据

2017年6月11日　　15:15

使用get方法请求数据:

GET /3.txt HTTP/1.1
Host: localhost:2222
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:24.0) Gecko/201001　01 Firefox/24.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-cn,zh;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
If-Modified-Since: Fri, 18 Jul 2014 08:36:36 GMT\r\n
空行
请求数据(可以为空)

使用post方法请求数据:

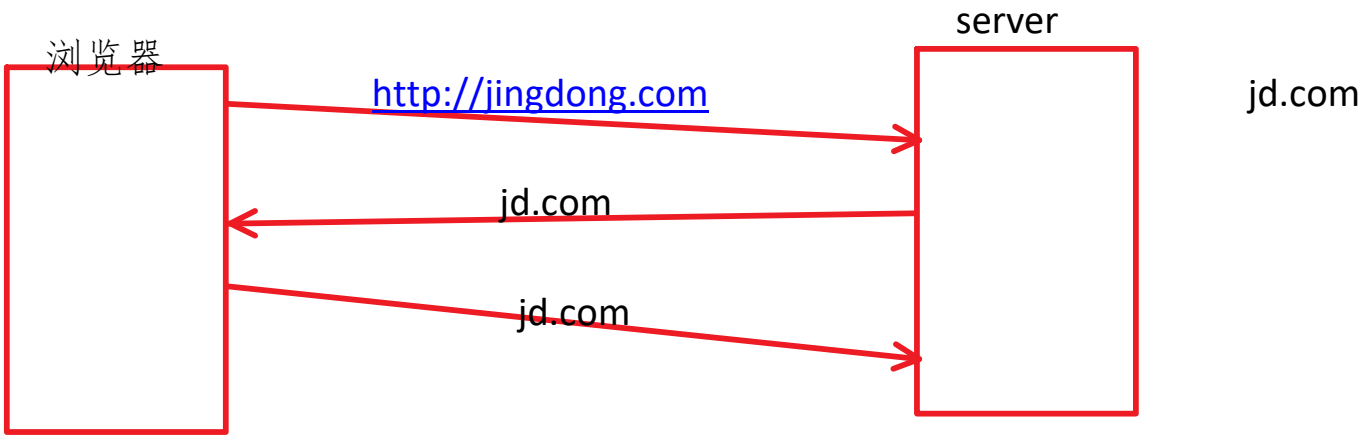
POST　HTTP/1.1
Host: localhost:2222
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:24.0) Gecko/201001　01 Firefox/24.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-cn,zh;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
If-Modified-Since: Fri, 18 Jul 2014 08:36:36 GMT
空行
user=詹姆斯&pwd=James&sex=男

浏览器地址栏:
192.168.1.115/hello.c
浏览器封装一个http请求协议

get /hello.c http/1.1
key:value
key:value
key:value
key:value
\r\n

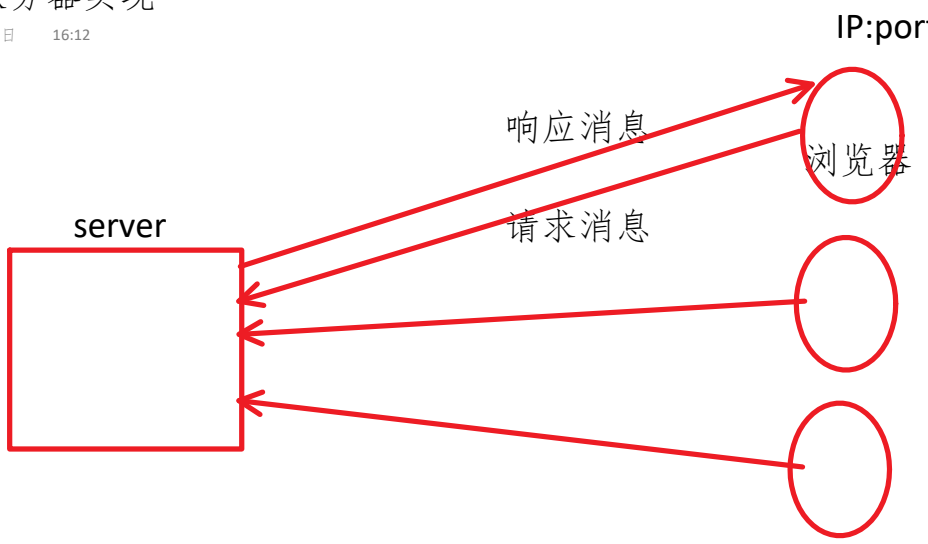
重定向

2017年6月11日　　15:44



web服务器实现

2017年6月11日 16:12



http
- 传输协议：tcp

服务器端的实现：

```
void http_respond_head(int cfd, char* type)
{
    char buf[1024];
    // 状态行
    sprintf(buf, "http/1.1 200 OK\r\n");
    write(cfd, buf, strlen(buf));
    // 消息报头
    sprintf(buf, "Content-Type: %s\r\n", type);
    write(cfd, buf, strlen(buf));

    // 空行
    write(cfd, "\r\n", 2);
}

void main()
{
    // 修改进程的工作目录
    chdir(path);
    // 创建监听的套接字
    int lfd = socket(af_inet, sock_stream, 0);
    // 绑定
    struct sockaddr_in serv;
    serv.family = af_inet;
    serv.port = htons(8989);
    bind(lfd, &serv, len);
    // 监听
    listen();

    int cfd = accept();
```

```
    //读数据
    read(cfd, buf, sizeof(buf));
    //先将buf中的请求行拿出来
    // GET /hello.c http/1.1
    char method[12], path[1024], protocol[12];
    //得到文件名
    char* file = path+1;
    //打开文件
    int fdd = open(file, O_RDONLY);
    int len = 0;
    http_respond_head(cfd, "text/plain");
    //循环读数据
    while( (len=read(fdd, buf, sizeof(buf))) > 0)
    {
        //数据发送给浏览器
        write(fdd, buf, len));
    }
}
```

== http中的文件类型

普通文件: text/plain; charset=utf-8
*.html : text/html; charset=utf-8
*.jpg : image/jpeg
*.gif : image/gif
*.png : image/png
*.wav : audio/wav
*.avi : video/x-msvideo
*.mov : video/quicktime
*.mp3 : audio/mpeg

QFile

charset=iso-8859-1 西欧的编码，说明网站采用的编码是英文；
charset=gb2312 说明网站采用的编码是简体中文；
charset=utf-8 代表世界通用的语言编码；
可以用到中文、韩文、日文等世界上所有语言编码上
charset=euc-kr 说明网站采用的编码是韩文；
charset=big5 说明网站采用的编码是繁体中文；

2Day

1. 编写函数解析http请求
    - GET /hello.html HTTP/1.1\r\n
    - 将上述字符串分为三部分解析出来
2. 编写函数根据文件后缀，返回对应的文件类型
3. sscanf - 读取格式化的字符串中的数据
    - 使用正则表达式拆分
    - [^]的用法
4. 通过浏览器请求目录数据
    - 读指定目录内容
        - opendir
        - readdir
        - closedir
    - scandir - 扫描dir目录下(不包括子目录)内容
5. http中数据特殊字符编码解码问题
    - 编码
    - 解码

recv的flag

2017年6月13日 10:42

```
n = recv(sock, &c, 1, MSG_PEEK);
```

- flag == MSG_PEEK
- recv从缓冲区总读数据-拷贝的方式

1234567890

recv(fd, buf, size, 0);

- 没数据了

recv(fd, buf, size, MSG_PEEK);

- 有,1234567890

正则表达式

http://deerchao.net/tutorials/regex/regex.htm

? http://www.jb51.net/tools/regexsc.htm

正则表达式速查表

字符	描述
\	将下一个字符标记为一个特殊字符、或一个原义字符、或一个向后引用、或一个八进制转义符。例如，“n"匹配字符"n"。"\n"匹配一个换行符。串行"\\"匹配"\"而"\("则匹配"("。
^	匹配输入字符串的开始位置。如果设置了RegExp对象的Multiline属性，^也匹配“\n"或"\r"之后的位置。
$	匹配输入字符串的结束位置。如果设置了RegExp对象的Multiline属性，$也匹配“\n"或"\r"之前的位置。
*	**匹配前面的子表达式零次或多次。例如，zo*能匹配“z"以及"zoo"。*等价于{0,}。**
+	匹配前面的子表达式一次或多次。例如，“zo+"能匹配"zo"以及"zoo"，但不能匹配"z"。+等价于{1,}。
?	匹配前面的子表达式零次或一次。例如，“do(es)?"可以匹配"does"或"does"中的"do"。?等价于{0,1}。
{n}	n是一个非负整数。匹配确定的n次。例如，“o{2}"不能匹配"Bob"中的"o"，但是能匹配"food"中的两个o。
{n,}	n是一个非负整数。至少匹配n次。例如，“o{2,}"不能匹配"Bob"中的"o"，但能匹配"foooood"中的所有o。"o{1,}"等价于"o+"。"o{0,}"则等价于"o*"。
{n,m}	m和n均为非负整数，其中n<=m。最少匹配n次且最多匹配m次。例如，“o{1,3}"将匹配"fooooood"中的前三个o。"o{0,1}"等价于"o?"。请注意在逗号和两个数之间不能有空格。
?	当该字符紧跟在任何一个其他限制符（*,+,?，{n}，{n,}，{n,m}）后面时，匹配模式是非贪婪的。非贪婪模式尽可能少的匹配所搜索的字符串，而默认的贪婪模式则尽可能多的匹配所搜索的字符串。例如，对于字符串“oooo"，"o+?"将匹配单个"o"，而"o+"将匹配所有"o"。
.	匹配除“\n"之外的任何单个字符。要匹配包括"\n"在内的任何字符，请使用像"(.|\n)"的模式。
(pattern)	匹配pattern并获取这一匹配。所获取的匹配可以从产生的Matches集合得到，在VBScript中使用SubMatches集合，在JScript中则使用$0…$9属性。要匹配圆括号字符，请使用“\("或"\)"。
(?:pattern)	匹配pattern但不获取匹配结果，也就是说这是一个非获取匹配，不进行存储供以后使用。这在使用或字符“(|)"来组合一个模式的各个部分是很有用。例如"industr(?:y|ies)"就是一个比"industry|industries"更简略的表达式。
(?=pattern)	正向肯定预查，在任何匹配pattern的字符串开始处匹配查找字符串。这是一个非获取匹配，也就是说，该匹配不需要获取供以后使用。例如，“Windows(?=95|98|NT|2000)"能匹配"Windows2000"中的"Windows"，但不能匹

	配"Windows3.1"中的"Windows"。预查不消耗字符，也就是说，在一个匹配发生后，在最后一次匹配之后立即开始下一次匹配的搜索，而不是从包含预查的字符之后开始。
(?!pattern)	正向否定预查，在任何不匹配pattern的字符串开始处匹配查找字符串。这是一个非获取匹配，也就是说，该匹配不需要获取供以后使用。例如“Windows(?!95|98|NT|2000)"能匹配"Windows3.1"中的"Windows"，但不能匹配"Windows2000"中的"Windows"。预查不消耗字符，也就是说，在一个匹配发生后，在最后一次匹配之后立即开始下一次匹配的搜索，而不是从包含预查的字符之后开始
(?<=pattern)	反向肯定预查，与正向肯定预查类拟，只是方向相反。例如，“(?<=95|98|NT|2000)Windows"能匹配"2000Windows"中的"Windows"，但不能匹配"3.1Windows"中的"Windows"。
(?<!pattern)	反向否定预查，与正向否定预查类拟，只是方向相反。例如“(?<!95|98|NT|2000)Windows"能匹配"3.1Windows"中的"Windows"，但不能匹配"2000Windows"中的"Windows"。
x|y	匹配x或y。例如，“z|food"能匹配"z"或"food"。"(z|f)ood"则匹配"zood"或"food"。
[xyz]	**字符集合。匹配所包含的任意一个字符。例如，“[abc]"可以匹配"plain"中的"a"。**
[^xyz]	负值字符集合。匹配未包含的任意字符。例如，“[^abc]"可以匹配"plain"中的"p"。
[a-z]	字符范围。匹配指定范围内的任意字符。例如，“[a-z]"可以匹配"a"到"z"范围内的任意小写字母字符。
[^a-z]	负值字符范围。匹配任何不在指定范围内的任意字符。例如，“[^a-z]"可以匹配任何不在"a"到"z"范围内的任意字符。
\b	匹配一个单词边界，也就是指单词和空格间的位置。例如，“er\b"可以匹配"never"中的"er"，但不能匹配"verb"中的"er"。
\B	匹配非单词边界。“er\B"能匹配"verb"中的"er"，但不能匹配"never"中的"er"。
\cx	匹配由x指明的控制字符。例如，\cM匹配一个Control-M或回车符。x的值必须为A-Z或a-z之一。否则，将c视为一个原义的“c"字符。
\d	**匹配一个数字字符。等价于[0-9]。**
\D	**匹配一个非数字字符。等价于[^0-9]。**
\f	匹配一个换页符。等价于\x0c和\cL。
\n	匹配一个换行符。等价于\x0a和\cJ。
\r	匹配一个回车符。等价于\x0d和\cM。
\s	匹配任何空白字符，包括空格、制表符、换页符等等。等价于[\f\n\r\t\v]。
\S	匹配任何非空白字符。等价于[^ \f\n\r\t\v]。
\t	匹配一个制表符。等价于\x09和\cI。
\v	匹配一个垂直制表符。等价于\x0b和\cK。
\w	匹配包括下划线的任何单词字符。等价于“[A-Za-z0-9_]"。
\W	匹配任何非单词字符。等价于“[^A-Za-z0-9_]"。
\x*n*	匹配*n*，其中*n*为十六进制转义值。十六进制转义值必须为确定的两个数字长。例如，“\x41"匹配"A"。"\x041"则等价于"\x04&1"。正则表达式中可以使用ASCII编码。.

\num	匹配*num*，其中*num*是一个正整数。对所获取的匹配的引用。例如，“(.)\1"匹配两个连续的相同字符。
\n	标识一个八进制转义值或一个向后引用。如果\n之前至少n个获取的子表达式，则n为向后引用。否则，如果n为八进制数字（0-7），则n为一个八进制转义值。
\nm	标识一个八进制转义值或一个向后引用。如果\nm之前至少有nm个获得子表达式，则nm为向后引用。如果\nm之前至少有n个获取，则n为一个后跟文字m的向后引用。如果前面的条件都不满足，若n和m均为八进制数字（0-7），则\nm将匹配八进制转义值nm。
\nml	如果n为八进制数字（0-3），且m和l均为八进制数字（0-7），则匹配八进制转义值nml。
\un	匹配n，其中n是一个用四个十六进制数字表示的Unicode字符。例如，\u00A9匹配版权符号（©）。

常用正则表达式

用户名	/^[a-z0-9_-]{3,16}$/
密码	/^[a-z0-9_-]{6,18}$/
密码2	(?=^.{8,}$)(?=.*\d)(?=.*\W+)(?=.*[A-Z])(?=.*[a-z])(?!.*\n).*$ (由数字/大写字母/小写字母/标点符号组成，四种都必有，8位以上)
十六进制值	/^#?([a-f0-9]{6}|[a-f0-9]{3})$/
电子邮箱	/^([a-z0-9_\.-]+)@([\da-z\.-]+)\.([a-z\.]{2,6})$/ /^[a-z\d]+(\.[a-z\d]+)*@([\da-z](-[\da-z])?)+(\.{1,2}[a-z]+)+$/或 \w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*
URL	/^(https?:\/\/)?([\da-z\.-]+)\.([a-z\.]{2,6})([\/\w \.-]*)*\/?$/ 或 [a-zA-z]+://[^\s]*
IP 地址	/((2[0-4]\d|25[0-5]|[01]?\d\d?)\.){3}(2[0-4]\d|25[0-5]|[01]?\d\d?)/ /^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$/ 或 ((2[0-4]\d|25[0-5]|[01]?\d\d?)\.){3}(2[0-4]\d|25[0-5]|[01]?\d\d?)
HTML 标签	/^<([a-z]+)([^<]+)*(?:>(.*)<\/\1>|\s+\/>)$/或<(.*)(.*)>.*<\/\1>|<(.*) \/>
删除代码\\注释	(?<!http:|\S)//.*$
匹配双字节字符(包括汉字在内)	[^\x00-\xff]
汉字(字符)	[\u4e00-\u9fa5]
Unicode编码中的汉字范围	/^[\u2E80-\u9FFF]+$/
中文及全角标点符号(字符)	[\u3000-\u301e\ufe10-\ufe19\ufe30-\ufe44\ufe50-\ufe6b\uff01-\uffee]
日期(年-月-日)	(\d{4}|\d{2})-((0?([1-9]))|(1[1|2]))-((0?[1-9])|([12]([1-9]))|(3[0|1]))
日期(月/日/年)	((0?[1-9]{1})|(1[1|2]))/(0?[1-9]|([12][1-9])|(3[0|1]))/(\d{4}|\d{2})
时间(小时:分钟, 24小时制)	((1|0?)[0-9]|2[0-3]):([0-5][0-9])

中国大陆固定电话号码	(\d{4}-|\d{3}-)?(\d{8}|\d{7})
中国大陆手机号码	1\d{10} 1[0-9]{10}
中国大陆邮政编码	[1-9]\d{5}
中国大陆身份证号(15位或18位)	\d{15}(\d\d[0-9xX])?
非负整数(正整数或零)	\d+
正整数	[0-9]*[1-9][0-9]*
负整数	-[0-9]*[1-9][0-9]*
整数	-?\d+
小数	(-?\d+)(\.\d+)?
空白行	\n\s*\r 或者 \n\n(editplus) 或者 ^[\s\S]*\n
QQ号码	[1-9]\d{4,}
不包含abc的单词	\b((?!abc)\w)+\b
匹配首尾空白字符	^\s*|\s*$
编辑常用	以下是针对特殊中文的一些替换(editplus) ^[0-9].*\n ^[^第].*\n ^[习题].*\n ^[\s\S]*\n ^[0-9]*\. ^[\s\S]*\n <p[^<>*]> href="javascript:if\(confirm\('(.*?)'\)\)window\.location='(.*?)'" <span style=".[^"]*rgb\(255,255,255\)">.[^<>]* <DIV class=xs0>[\s\S]*?</DIV>

函数描述: 读取格式化的字符串中的数据。

函数原型:

```
int sscanf(
   const char *buffer,
   const char *format, [ argument ] ...
);
```

1. 取到**指定字符为止的字符串**。如在下例中，取遇到空格为止字符串。

```
1 sscanf("123456 abcdedf", "%[^ ]", buf);
2 printf("%s\n", buf);
```

结果为：**123456**

2. 取**仅包含指定字符集的字符串**。如在下例中，取仅包含1到9和小写字母的字符串。

```
1 sscanf("123456abcdedfBCDEF", "%[1-9a-z]", buf);
2 printf("%s\n", buf);
```

结果为：**123456abcdedf**

3. 取到**指定字符集为止的字符串**。如在下例中，取遇到大写字母为止的字符串。

```
1 sscanf("123456abcdedfBCDEF", "%[^A-Z]", buf);
2 printf("%s\n", buf);
```

结果为：**123456abcdedf**

```
#include <dirent.h>

int scandir(const char *dirp,
      struct dirent ***namelist,
      int (*filter)(const struct dirent *),
      int (*compar)(const struct dirent **,
            const struct dirent **));
```

dirp
- 当前要扫描的目录

namelist
- struct dirent** ptr;
- struct dirent* ptr[];
- &ptr;

filter
- NULL

compar
- 文件名显示的时候,指定排序规则
  - alphasort
  - versionsort

strftime 函数

头文件: time.h
函数功能: 将时间格式化，或者说格式化一个时间字符串。
函数原型:

```
size_t strftime(
    char *strDest,
    size_t maxsize,
    const char *format,
    const  struct tm *timeptr
);
```

- format
  - %a 星期几的简写
  - %A 星期几的全称
  - %b 月份的简写
  - %B 月份的全称
  - %c 标准的日期的时间串
  - %C 年份的前两位数字
  - %d 十进制表示的每月的第几天
  - %D 月/天/年
  - %e 在两字符域中，十进制表示的每月的第几天
  - %F 年-月-日
  - %g 年份的后两位数字，使用基于周的年
  - %G 年份，使用基于周的年
  - %h 简写的月份名
  - %H 24小时制的小时
  - %I 12小时制的小时
  - %j 十进制表示的每年的第几天
  - %m 十进制表示的月份
  - %M 十时制表示的分钟数
  - %p 本地的AM或PM的等价显示
  - %r 12小时的时间
  - %R 显示小时和分钟：hh:mm
  - %S 十进制的秒数

- %t 水平制表符
- %T 显示时分秒：hh:mm:ss
- %u 每周的第几天，星期一为第一天 （值从1到7，星期一为1）
- %U 第年的第几周，把星期日作为第一天（值从0到53）
- %V 每年的第几周，使用基于周的年
- %w 十进制表示的星期几（值从0到6，星期天为0）
- %W 每年的第几周，把星期一做为第一天（值从0到53）
- %x 标准的日期串
- %X 标准的时间串
- %y 不带世纪的十进制年份（值从0到99）
- %Y 带世纪部分的十制年份
- %z，%Z 时区名称，如果不能得到时区名称则返回空字符。

数据转码

url在数据传输过程中不支持中文，需要转码。

- 汉字
- 特殊字符
    - 查看manpage
        - man ascii
    - 要处理可见字符
        - 从space开始 - 32
        - 前0-31个不可见
    - 不需要转换的特殊字符
        - .
        - _
        - *
        - /
        - ~
        - 0-9
        - a-z
        - A-Z
    - 需要转换的字符使用其16进制的值前加%表示

可以在shell下通过unicode工具查看

安装unicode

- sudo apt-get install unicode