tinyHttp学习计划

# 学习目的

## 1.1.了解socket通信机制

## 1.2.了解http协议

## 1.3.cgi（是否可以去掉）

# 2.学习环境

Windows10 专业版

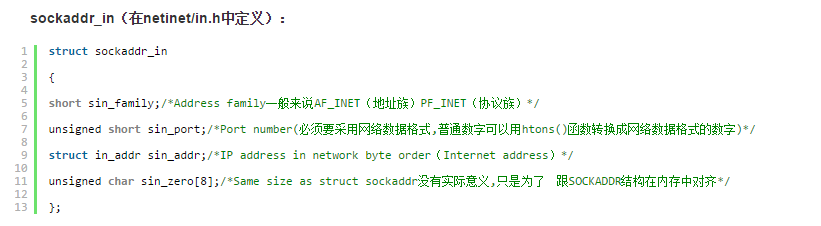
Visual Studio 2013

# 3.学习过程

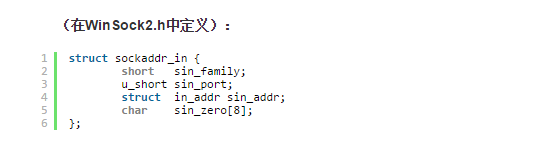
## 3.1结构定义

### 3.1.1sockaddr\_in

Linux系统下定义：



Winodows 系统下定义：



U\_short与unsigned short等价 二者差别就是 sin\_zero有无符号

### 3.1.2socklen\_t

windows平**台下**:  
头**文件**：#include<ws2tcpip.h>  
  
linux平**台下**：  
下面两个头**文件**都有**定义**：  
1）#include <sys/socket.h>  
2）#include <unistd.h>

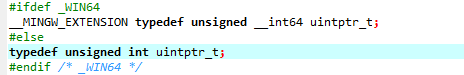
详细**定义**：typedef int socklen\_t;

### 3.1.3 pthread\_t

Window下：

头文件:#include <pthread.h>





Linux下:

typedef unsigned long int pthread\_t;

作用：用于声明线程ID。

## 3.2方法api

### 3.2.1 socket

httpd = socket(PF\_INET, SOCK\_STREAM, 0);// 创建套接口，由系统调用socket实现

\*AF\_INET 协议族 设置网络通信的域 值2 表示IPv4 Internet协议

\*SOCK\_STREAM 协议类型 设置套接字通信的类型 值1 Tcp连接，提供序列化的、可靠的、双向连接的字节流。支持带外数据传输

\* 0 制定某个协议的特定类型 。通常某协议中只有一种特定类型，这样protocol参数仅能设置为0；但是有些协议有多种特定的类型，就需要设置这个参数来选择特定的类型。



### 3.2.2 memset

void \*memset(void \*s, int ch, [size\_t](http://baike.baidu.com/view/3236587.htm) n);

函数解释：将s中当前位置后面的n个字节 （typedef unsigned int size\_t ）用 ch 替换并返回 s 。

memset：作用是在一段内存块中填充某个给定的值，它是对较大的[结构体](http://baike.baidu.com/view/204974.htm)或[数组](http://baike.baidu.com/view/209670.htm)进行清零操作的一种最快方法

参考：<http://baike.baidu.com/link?url=0vQD-7JOJ49EzmCOft4QBbsVHbOLvgf8Y4aJZdkvTClc-c6pjEzAFcrSp5GBW_tz9phc72Y88hUezmGOYqDXIq>

### 3.2.3.bind

**int** bind(**int** ***sockfd***, **const struct sockaddr** \****addr***,**socklen\_t** \****addrlen***);

sockfd是创建的一个套接字，其在网络中存在，但没有任何地址，这里就是将其与指定的地址绑定

参考: <http://www.cnblogs.com/java20130726/archive/2011/12/21/3218629.html>

### 3.2.4getsocktname

int getsockname(int sock, struct sockaddr \*name, socklen\_t \*namelen);

获取套接字ip地址

参考资料

<http://baike.baidu.com/link?url=_w2e9uynPVRy8QLWYQMSRjxDuu4Weuuy0pKIpiTcpNo7ug-w1X8-spoaplm5JBV8JcuLtl6GSsnx3z-V-DaG4faSLe6TtuMkaPv4CjZfT2i>

### 3.2.5 ntohs

将一个16位数由网络字节顺序转换为主机字节顺序

参考资料：

<http://baike.baidu.com/link?url=DbKuya89sgOM7gImjGnGp12fghoQvCaNCbIaQ67SEuS0_uRppbf90g0zsRc0wBKbxcV92V53nknegZIoOHJ4wEz3Q7l3b0aseoIQZB7IjqTopKey58SM87wxfSk8qUSf>

### 3.2.6 listen

int listen(int s, int backlog);

作用：等待连接

参考资料：<http://c.biancheng.net/cpp/html/364.html>

### 3.2.7 isspace

若参数c 为空白字符，则返回非 0，否则返回

参考资料：<http://c.biancheng.net/cpp/html/120.html>

### 3.2.8strcmp

作用:忽略字符串大小写

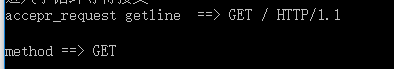
在Linux下方法名 叫做strcasecmp

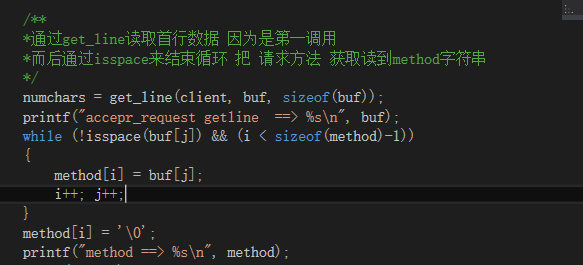
参考资料: <http://blog.csdn.net/naturebe/article/details/7437476>

## 3.3功能实现

### 3.3.1获取请求方法







Method就是获取到首行的空格进行截断

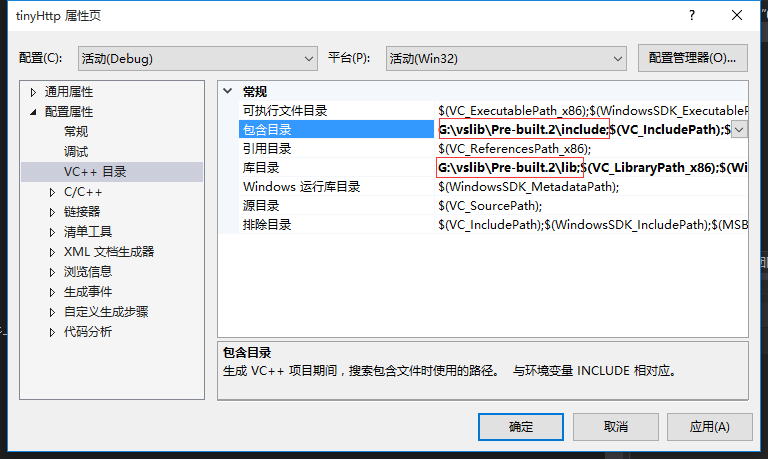
# 4.相关问题

## 4.1缺少#include <pthread.h>头文件

Step1:下载pthread.zip压缩包

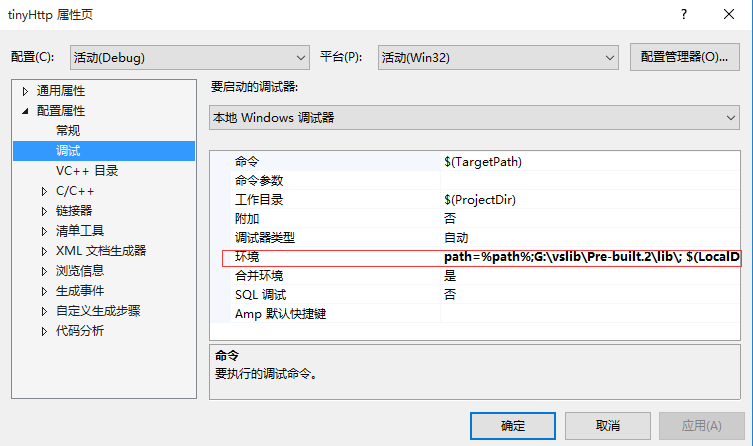
链接: https://pan.baidu.com/s/1boBGCZt 密码: x6hm里面存在，在我对应的文件夹里面也存在

Step2:点击 项目 -> 属性->vc++目录，配置路径，将库目录中添加指向Pre-built.2这个文件夹



Step3: 点击调试->环境

输入值 path=%path%;G:\vslib\Pre-built.2\lib\; $(LocalDebuggerEnvironment)



## 4.2无法查找或打开 PDB 文件。

参考资料：<http://jingyan.baidu.com/article/90bc8fc87fbafff653640ca6.html>

# 5个人疑问

## 5.1端口问题（未解决）

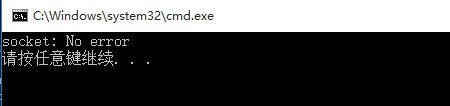
\*port = ntohs(name.sin\_port); 只是将网络字节顺序转化为主机字节顺序，不知是如何动态分配端口

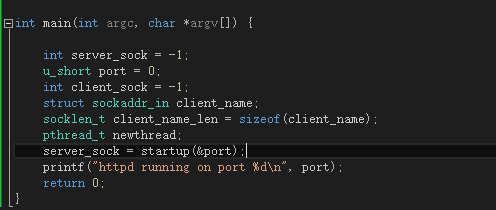
## 5.2listen不挂起（待确认）

在我写代码的时候发现运行 listen 运行后线程会退出，讲道理它是一个监听线程，等待连接，难道不应该挂起吗？

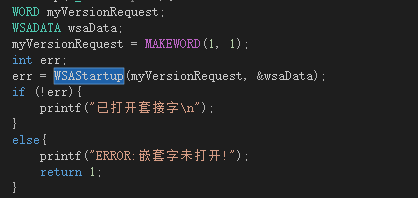
百度：listen函数不会block，只有在你accept的时候才会

## 5.3socket:no Error（已解决）





解决方案：在建立套接字连接之前，加入如下代码



WSAStartup 对Winsock服务的初始化

参考资料：

<http://baike.baidu.com/link?url=Lut36UXYJDa0KOrqtUl0LDKKj_HeBG7yR2C-I6nUolLx8JyNAT1mVlr3cv9QVtBGY_s8Ed0ggEnkQbRjasmiTJTYTCBUYyWhkmgBZAg4ud3>

## 5.4一次请求，建立两次连接（未解决）

在监听端口打开之后，向浏览器输入请求，connect连接进行了两次

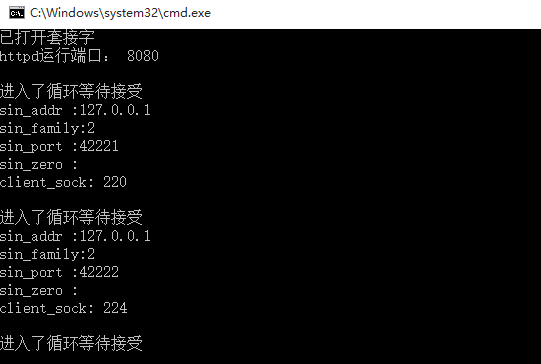
操作如下：



此时服务器端属于监听状态，然后我来进行请求

请求url : 127.0.0.1:8080(这是服务器端口)

而后控制输出信息:



Aceept执行了两次，很明显客户端连接了两次，不明白这是为什么，起初以为这是超时重传，但抓包后我发现并不是这样

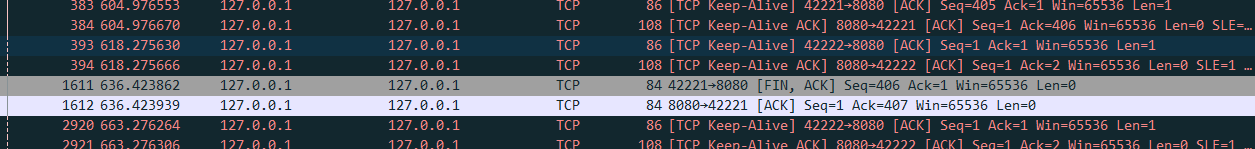


很明显 根据抓包数据可以知道两次连接都是成功，并且第二次连接还发出了释放连接包，关闭连接，可以看其状态处于终止等待状态



而且根据图可以看出 其依旧在发心跳包

过一段时间后：主动关闭客户端请求，很明显发现也会发出释放连接请求



此时状态





很明显服务端已经关闭这次连接 但是客户端没有，很明显是因为服务端没有发出连接释放报文段

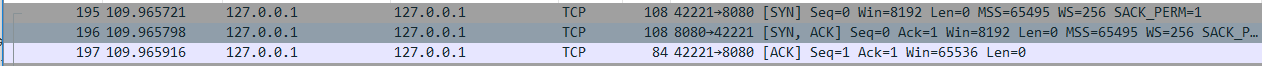
问题1：为什么会进行两次（已解决）

解答：因为涉及到浏览器的同源策略，在连接建立之后，会另外建立一个连接来获取服务器允许通过哪些方法进行处理(http一种有包括get post共八种方法) 进行确认之后 才会发送数据



执行过程

step1.建立正常请求连接



Step2:是获取访问资格



step3.发送请求数据



step4.服务器响应



很显然响应的只是客户端请求的数据，服务端没有返回任何数据，可见在执行accept后不做任何处理，依旧会响应客户端

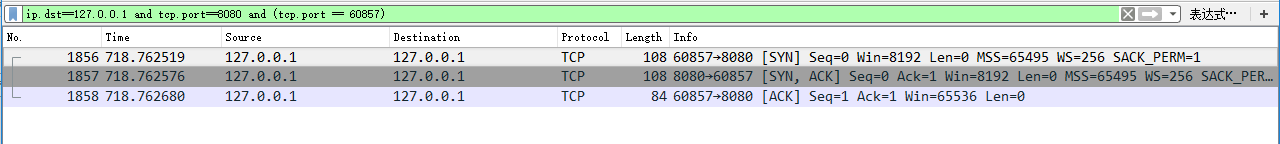
Step5关闭请求方法端口



根据上述问题 我不使用浏览器访问 看看效果







关闭telnet



问题2：第二次连接，客户端立即发送了连接释放连接报文段 此时服务端关闭 但是为什么没有发送释放报文端给客户端 确使其处于FIN\_WAIT\_2状态

服务器运行一段时间后，关闭了自己的socket，才会发送FIN

问题3：为什么连接的时候服务端会发送数据，关闭时却不释放

猜测：连接时候通过accept方法 释放连接却是之后的操作 并没有写

问题4：为什么始终会发送心跳包，并且服务端服务端会返回心跳包

答：因为客户端连接始终是未关闭，所以发送给服务端心跳包，检测是否后还连接

但是为什么此时服务器端也会发送心跳包，通过什么处理的。

## 5.5error LNK2019:外部符号 \_\_imp\_\_pthread\_create

这个需要引入线程的lib文件

参考资料： <http://blog.csdn.net/HangloveJW1314/article/details/51888309>

# 6.扩展学习

## 6.1术语解释

### 6.1.1SYN

TCP连接的第一个包，非常小的一种数据包

参考资料：

<http://baike.baidu.com/link?url=yp5PrOf1stJlAJX6jzpGYtK6V2Ysh8O-43TgsZpW71EMI2Uic-Av5y_CExBrU-dvsf-qsEE-OAHxnOF-0-__xK>

### 6.1.2ACK

表示发来的数据已确认接收无误

参考资料：

<http://baike.baidu.com/item/ACK?sefr=enterbtn>

### 6.1.3 FIN\_WAIT\_2

参考资料: <http://www.2cto.com/shouce/ApacheManual/misc/fin_wait_2.html>

## 6.2网络连接

### 6.2.1http连接方法

参考资料 <http://blog.csdn.net/truong/article/details/19936541>

### 6.2.2浏览器请求过程

参考资料 <http://www.jianshu.com/p/558455228c43>

### 6.2.3同源策略

同源策略详解及绕过（Part1）

参考资料： <http://www.freebuf.com/articles/web/65468.html>

浏览器和服务器实现跨域(CORS)判定的原理

参考资料： <http://blog.csdn.net/zmx729618/article/details/53319383>

https options

参考资料: <http://blog.csdn.net/leikezhu1981/article/details/7402272>

http深入浅出

参考资料：<http://www.cnblogs.com/yin-jingyu/archive/2011/08/01/2123548.html>

### 6.2.4TCP建立，释放连接

参考资料： <http://blog.chinaunix.net/uid-26413668-id-3376762.html>

### 6.2.5TCP、UDP、IP 协议分析

参考资料：<http://blog.chinaunix.net/uid-26833883-id-3627644.html>

# 7.参考文献