**北京邮电大学课程设计报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **课程设计**  **名称** | 计算机网络课程设计 | | **学 院** | 计算机 | **指导教师** | **蒋砚军老师** |
| **班 级** | **班内序号** | **学 号** | | **学生姓名** | **成绩** | |
| **2020211305** |  | **2020211376** | | **马天成** |  | |
| **2020211305** |  | **2020211383** | | **王宸** |  | |
| **2020211305** |  | **2020211409** | | **罗帅** |  | |
| **课**  **程**  **设**  **计**  **内**  **容** | 一、课程设计教学目的：把计算机网络中所学的DNS的理论知识与实际编程联系起来，加强学生对协议的了解、思考能力和编程能力，培养团队合作和沟通能力。  二、基本内容：设计一个DNS服务器程序，读入“域名-IP地址”对照表，当客户端查询域名对应的IP地址时，用域名检索该对照表，三种检索结果： 检索结果为ip地址0.0.0.0，则向客户端返回“域名不存在”的报错消息，而不是返回IP地址为0.0.0.0（不良网站拦截功能）。 检索结果为普通IP地址，则向客户返回这个地址（服务器功能）。 表中未检到该域名，则向因特网DNS服务器发出查询，并将结果返给客户端（中继功能）。  三、实验方法：明确流程进行顺序、程序模块划分和函数接口，合作编写程序、测试及调试程序，完成课程设计；  四、团队分工： 马天成：参与初期设计、编写本地映射表和chche的构造、检索、更新功能、改进程序的结构主体，进行程序调试。 罗帅：参与初期设计、编写报文的解析读取信息和响应报文的重构功能、完善程序的结构主体，参与程序调试。 王宸：参与初期设计、编写程序主体框架和socket通信配置初始化，完善程序的结构主体，参与程序调试。 | | | | | |
| **学生课程设计报告**  （附页） |  | | | | | |
| **课程设计**  **成绩评定** | **评语**:  **成绩**:  指导教师签名：  年 月 日 | | | | | |

注：评语要体现每个学生的工作情况，可以加页。

[一：实验目的 3](#_Toc107783073)

[二：实验环境 3](#_Toc107783074)

[三：实验内容详解 3](#_Toc107783075)

[3.1 功能设计 3](#_Toc107783076)

[3.2 全局模块 4](#_Toc107783077)

[3.3 header 5](#_Toc107783078)

[3.4 package 5](#_Toc107783079)

[3.4.1 报文格式说明 5](#_Toc107783080)

[3.4.2 报文处理函数 6](#_Toc107783081)

[3.5 cache 9](#_Toc107783082)

[3.5.1 数据结构 9](#_Toc107783083)

[3.5.2 Cache函数 9](#_Toc107783084)

[3.6 ID 11](#_Toc107783085)

[3.6.1 数据结构 11](#_Toc107783086)

[3.6.2 ID函数 11](#_Toc107783087)

[3.7 dnsTire 13](#_Toc107783088)

[3.7.1 数据结构 13](#_Toc107783089)

[3.7.2 dnsTire函数 13](#_Toc107783090)

[3.8 主函数模块 15](#_Toc107783091)

[3.8.1 准备阶段 15](#_Toc107783092)

[3.8.2 工作阶段 16](#_Toc107783093)

[四：测试与调整 18](#_Toc107783094)

[4.1 Cache机制调整 18](#_Toc107783095)

[4.2 Cache容量参数调整 18](#_Toc107783096)

[4.3 ID表的机制调整 18](#_Toc107783097)

[4.4 ID表的容量调整 18](#_Toc107783098)

[4.5 各种宏定义参数的设置 18](#_Toc107783099)

[4.6 package处理问题 19](#_Toc107783100)

[4.7 外部发送和Ipv6引出的屏蔽机制升级 19](#_Toc107783101)

[4.8 遇到的小问题 19](#_Toc107783102)

[五：正式测试结果及分析 19](#_Toc107783103)

[5.1 本机测试 19](#_Toc107783104)

[5.2 外部主机测试 20](#_Toc107783105)

[5.3 屏蔽外部DNS服务器测试核心功能 21](#_Toc107783106)

[5.3.1 屏蔽功能 21](#_Toc107783107)

[5.3.2 本地查询功能 22](#_Toc107783108)

[5.3.3 外部查询功能&突发式查询测试 22](#_Toc107783109)

[5.4 跨平台机制 23](#_Toc107783110)

[5.5 字典接口 24](#_Toc107783111)

[5.6 LRU缓冲池 24](#_Toc107783112)

[六：实验总结 24](#_Toc107783113)

[6.1 实验经历 24](#_Toc107783114)

[6.2 实验感想 24](#_Toc107783115)

# 一：实验目的

设计一个DNS服务器程序，读入“域名-IP地址”对照表，当客户端查询域名对应的IP地址时，用域名检索该对照表，三种检索结果：

* 检索结果为IP地址0.0.0.0，则向客户端返回“域名不存在”的报错消息（不良网站拦截功能）；
* 检索结果为普通IP地址，则向客户返回这个地址（服务器功能）；
* 表中未检到该域名，则向因特网DNS服务器发出查询，并将结果返给客户端（中继功能）；

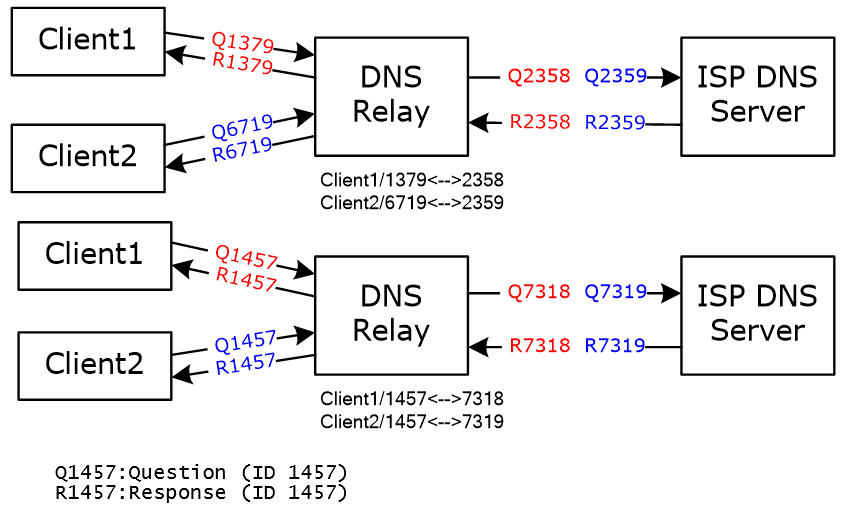
此外，虑多个计算机上的客户端会同时查询，需要进行消息ID的转换。

# 二：实验环境

1. 操作环境：windows
2. 编译软件：vs2020
3. 程序语言：c语言

# 三：实验内容详解

## 3.1 功能设计



采用模块化设计。主要分为以下几个模块：

* DNS.c：主函数模块，进行socket的绑定；收发流程和各类模块功能的调用
* global：（分为.h&.c）全局模块，进行一些通用的宏定义；定义时间计数线程函数
* header.h：头部报文结构体。（参考PPT）
* package：（分为.h&.c）包处理模块，有取出Url，ip以及构造报文等
* cache：（分为.h&.c）cache模块，各类查找更新机制
* ID：（分为.h&.c）ID转换表，核心转换机制
* dnsTire：（分为.h&.c）字典树，使用域名针对本地进行快速查询

主函数模块调用其他模块进行功能实现。

## 3.2 全局模块



global.h

* 全局变量

level：调试等级

timeCircle：时间循环，一秒一条，0-999

* 宏定义

ASCIISIZE：ascii码的规模，在字典树作为子树数量使用

URLMAXSIZE：Url（域名）的最大长度

IPMAXSIZE：IP（IP地址）的最大长度

LEN：package的最大长度

TIMEMODE：时间的轮回限制，如同因特网的12000使用

* 时间函数



**global.c**

timeCircle计数：歇一秒跳一下。数值为0-999

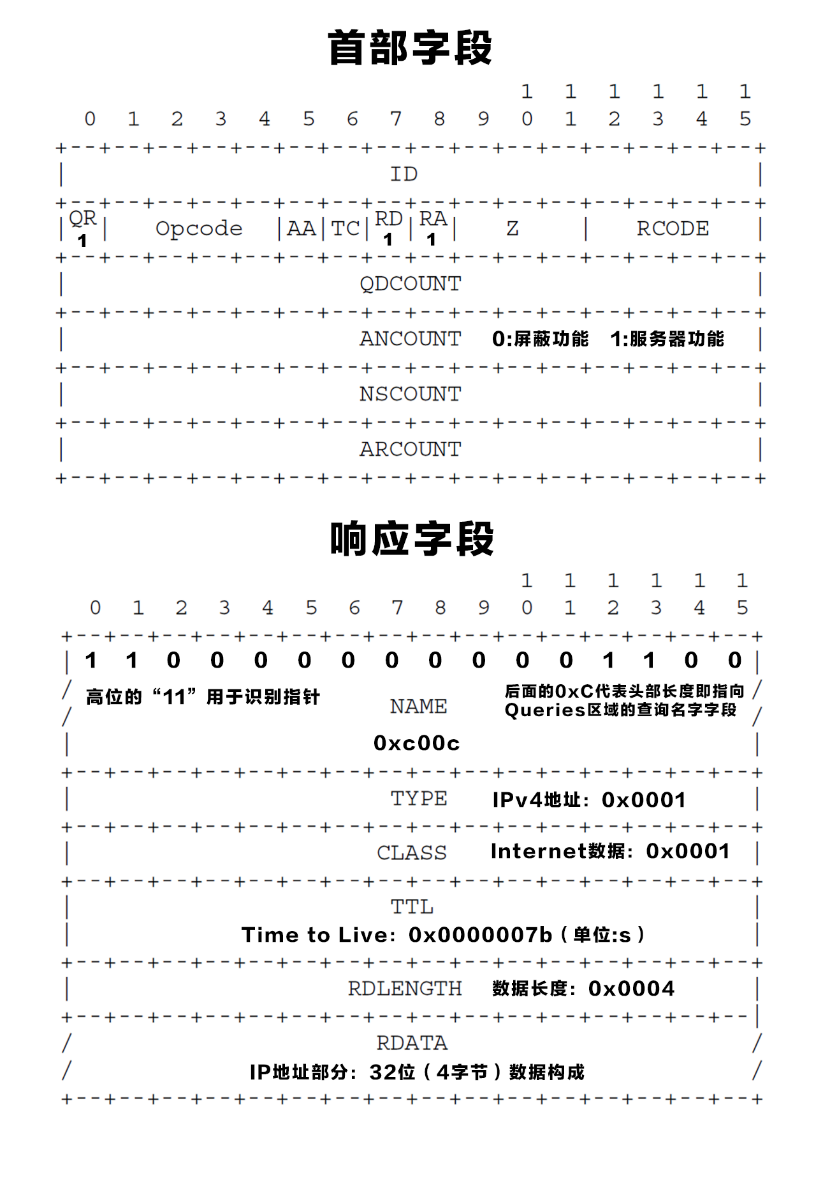
## 3.3 header



**header.h**

## 3.4 package

### 3.4.1 报文格式说明





* 标识

即ID字段，用于区分不同的报文。

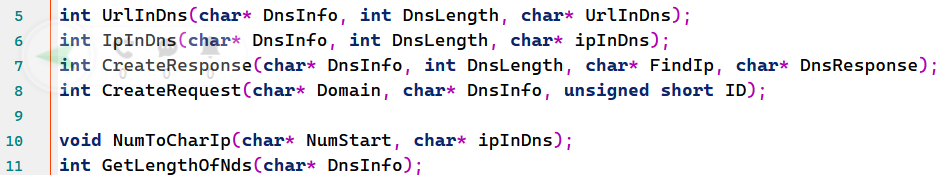
这个ID对于客户端是唯一的，但对于中继器和服务器是不唯一的。

* 标志

对于不同的报文类型，对应不同的数值，如请求报文为0001x；

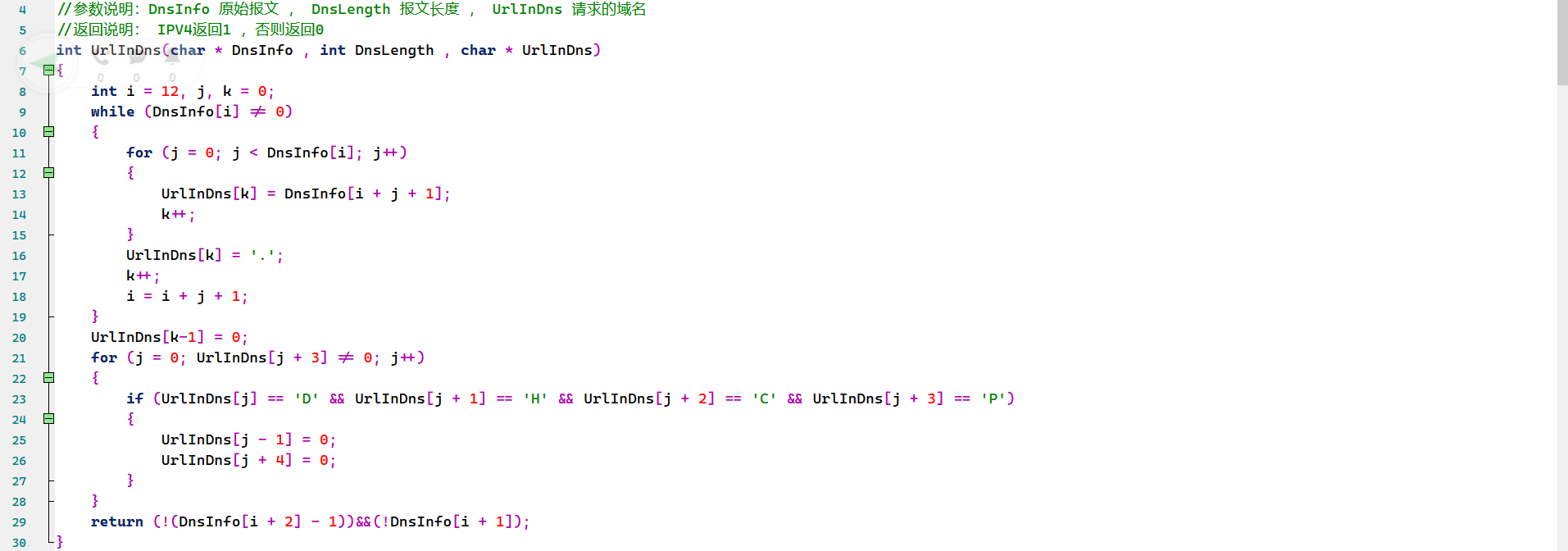
（。。。。补充一下其他的段，尽量写全）

### 3.4.2 报文处理函数



package.h

1. **在dns报文中取出URL**



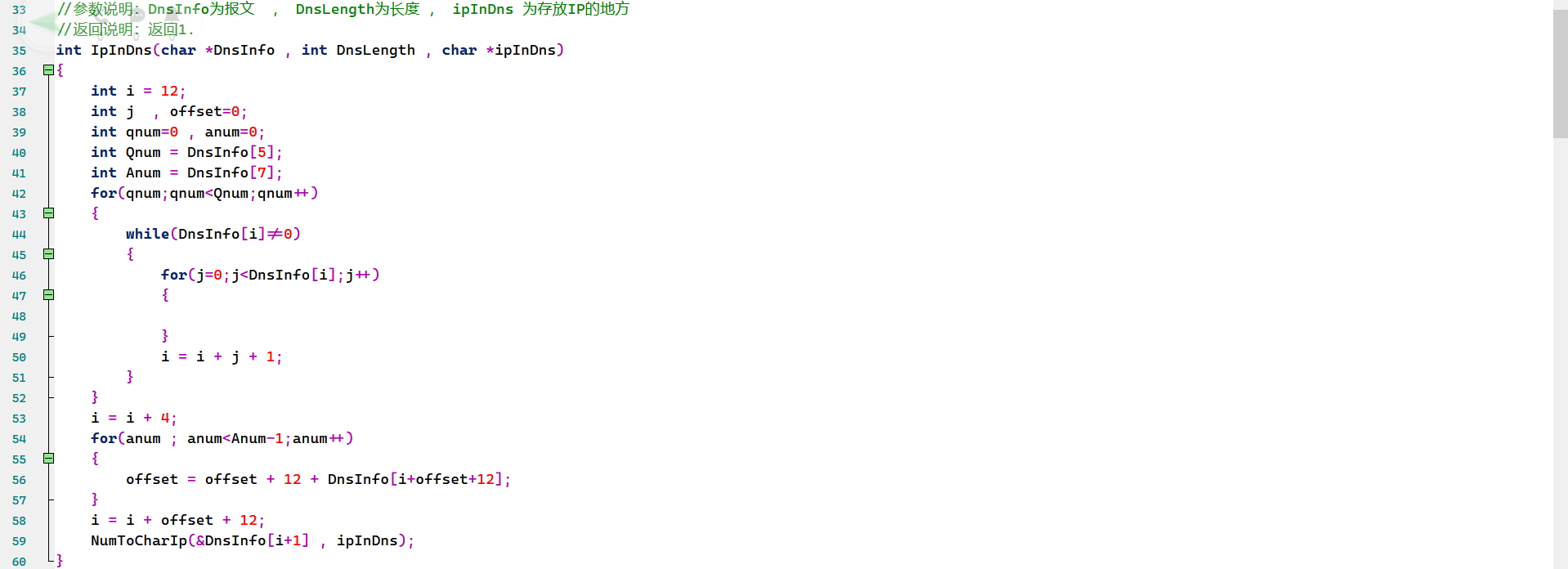
**int UrlInDns(char \* DnsInfo , int DnsLength , char \*UrlInDns);**

1. 参数说明：DnsInfo :原始报文 ， DnsLength : 原始报文长度 ，UrlInDns:域名

2. 返回说明：ipv4返回1，否则返回0

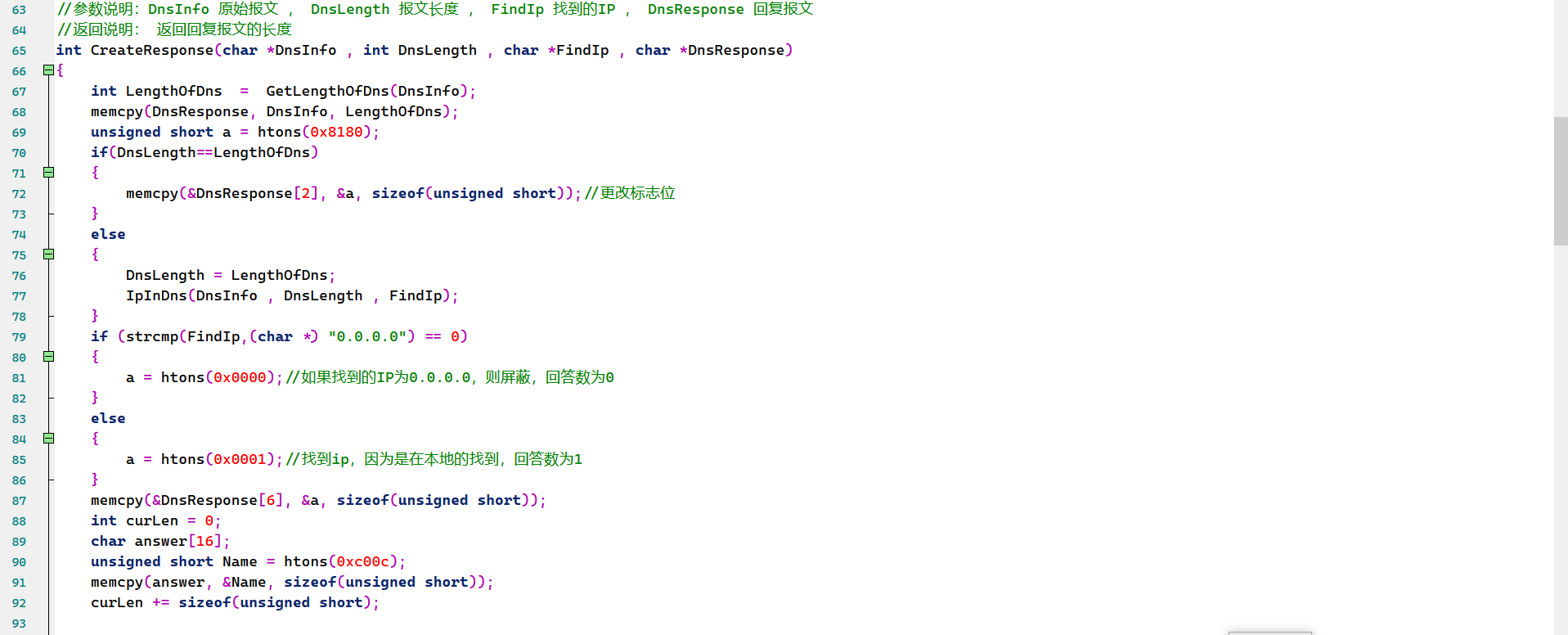
3. 函数说明：根据传入的参数，将报文中的域名提取出来，放在UrlInDns中

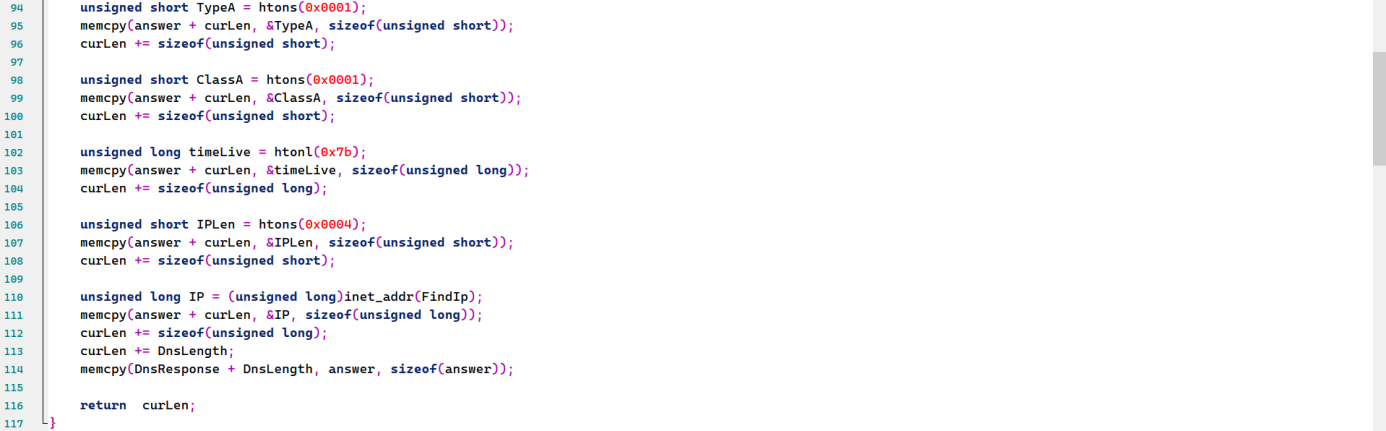
1. **在dns报文中取出IP**



**int IpInDns(char \*DnsInfo , int DnsLength ,char \*ipInDns)**1. 参数说明：DnsInfo:原始DNS报文  ，DnsLength:报文长度 ，ipInDns:存放报文中的IP   
2. 返回说明：返回构造的ip的长度  
3. 函数说明：根据原始报文 ， 将该报文中的ip地址提取出来，以此填充cache

1. **创建回应报文**





**int CreatResponse(char \*DnsInfo ,int DnsLength ,char \* FindIp ,  char \* DnsReponse)**

1. 参数说明

DnsInfo :原始报文 ， DnsLength : 原始报文长度 ，FindIp:响应报文的 IP , DnsReponse：响应报文

2. 返回说明：返回构造的响应报文的长度  
3. 函数说明：根据传入的原始报文，和找到的IP，构造出对应的响应报文，并且在函数内部实 现屏蔽功能

1. **创建请求报文**



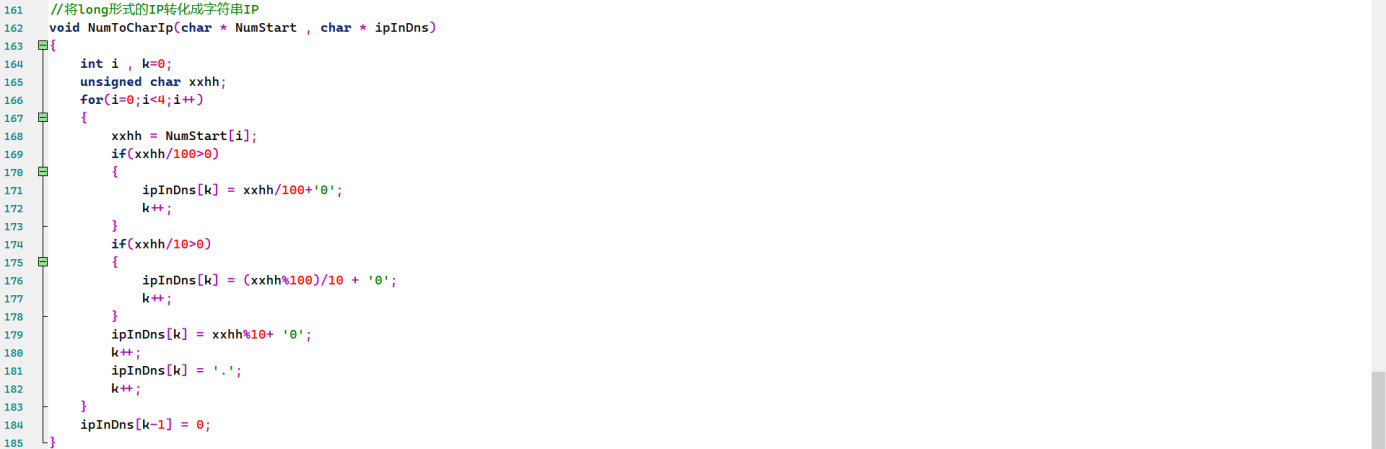
**int CreateRequest(char \*Domain , char \*DnsInfo , unsigned short ID)**

1. 参数说明：Domain:构造请求报文的域名 ，  ID：该请求包的ID ， DnsInfo：存放构造的请求报文

2. 返回说明：返回构造的请求报文的长度

3. 函数说明：根据传入的域名和ID，构造出询问该域名的报文，并且ID为ID转化表中的ID

**5. long型数字转化为IP**

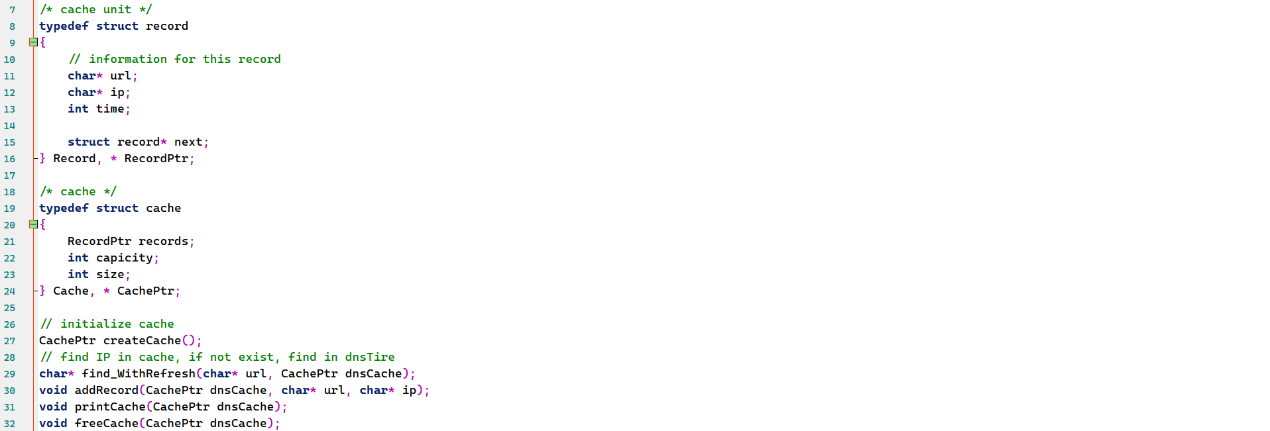


**6. 获得dns报文长度**



## 3.5 cache

### 3.5.1 数据结构



* Record

每一个Cache记录，有url，ip和time（进入的时间，timeCircle赋值）三个属性。

因为链表结构，所以有指向下一个的指针。

* Cache

使用链表数据结构，因为有LRU机制，需要经常性的替换和增删。

链表新纪录存储在头部，越往下时间越靠近更新的机制。便于更新。

Records数组，有头部空链表。

size控制当前记录数量，capicity控制容量。

* 宏定义



**CAPICITY**：容量限制

**REFRESHSIZE**：一般在添加时先判断在头部此规模中是否在Cache已有。有则更新至最顶端，无需添加。测试后选中此值。

**OVERTIME**：Cache的更新时间。超过这个时间记录视为超时。需要在扫描过后进行删除。

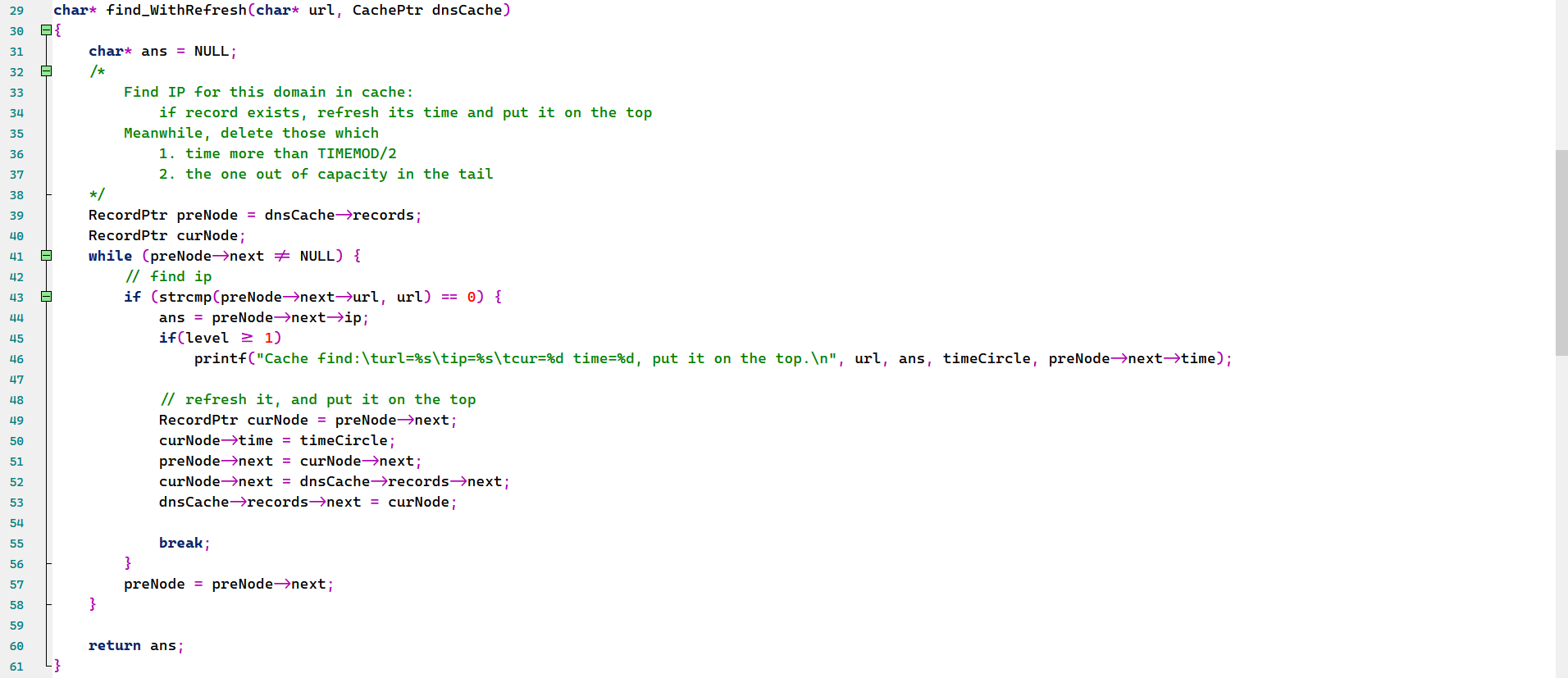
### 3.5.2 Cache函数

**1. 创建cache**



主要是建立cache里的record链表头部，并初始化容量和个数。返回值为申请的cache指针。

**2. 根据URL查找对应的IP记录，并进行刷新**



* 返回结果

如果找到，则返回对应的指针。（该指针已指向一片空间）

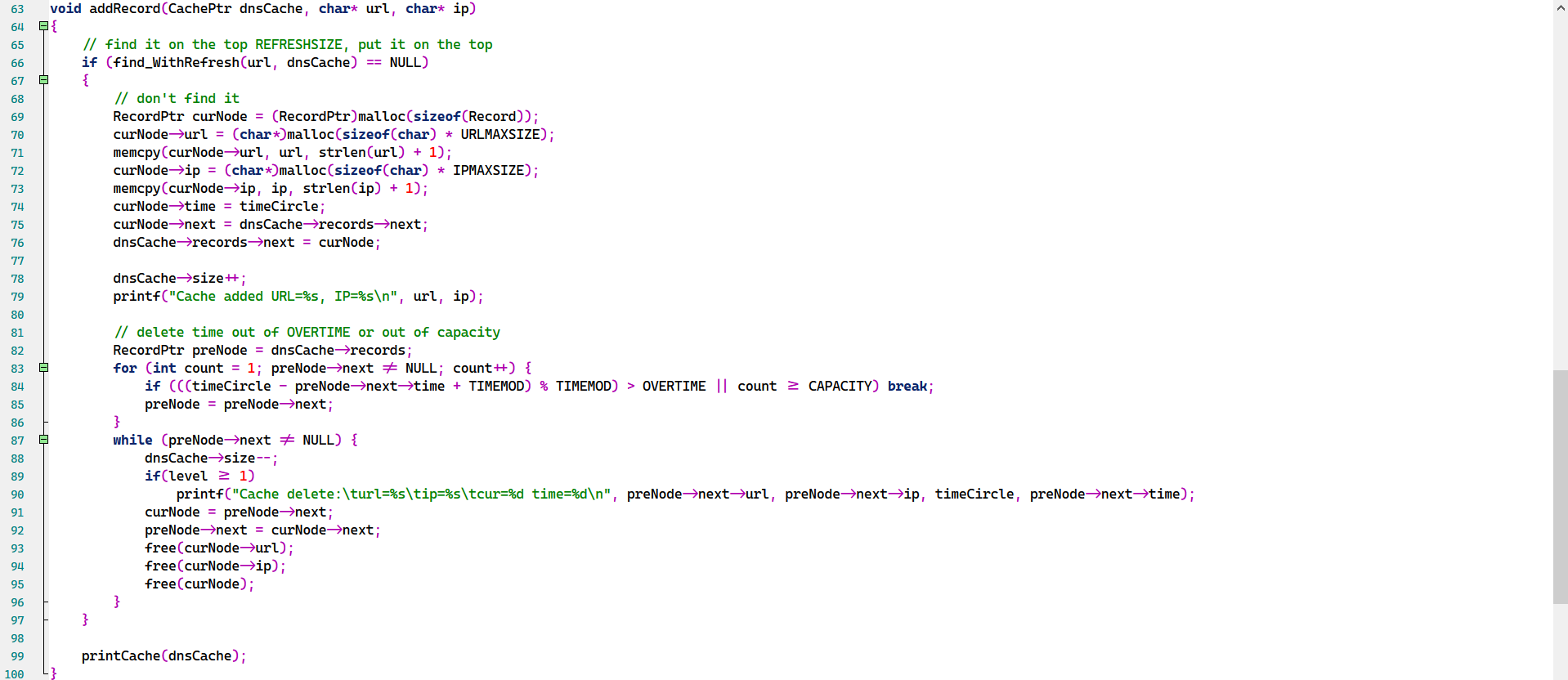
如果没有找到，则返回NULL。

* 算法设计

从头部链表开始找，找到则返回，找不到就不返回；

如果找到，那么刷新这条记录，放到cache最顶端。

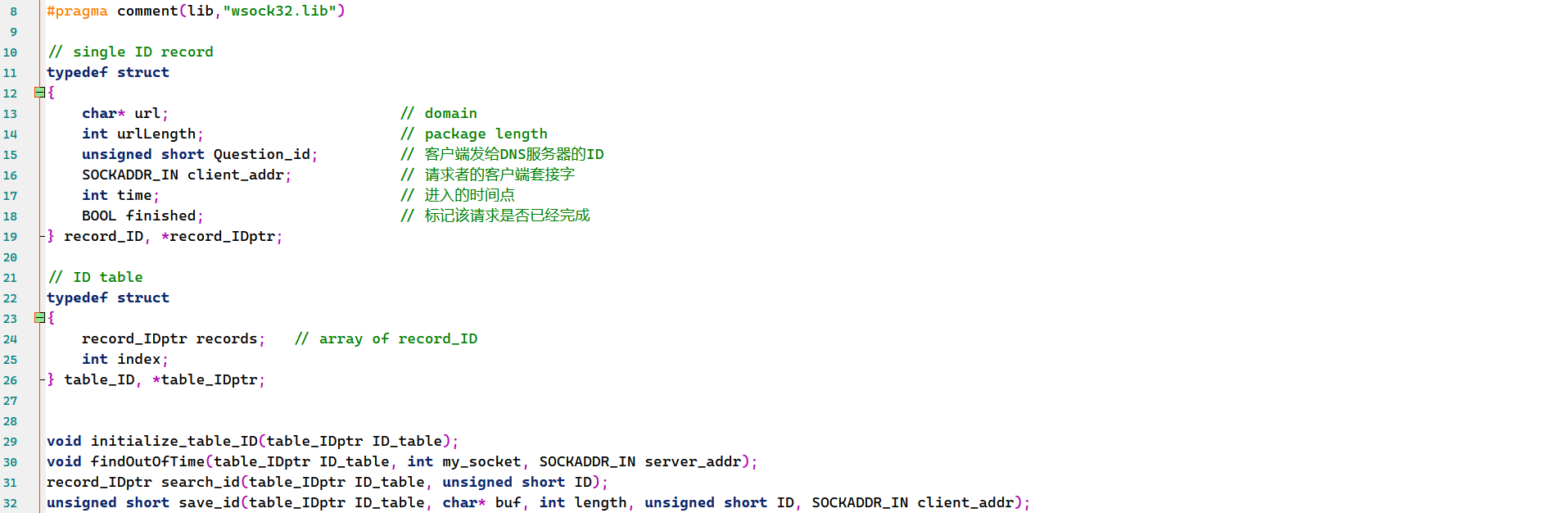
**3. 在cache中添加URL与IP**



* 对于每一次cache插入，加入先行寻找刷新机制，根据找到了记录就放至顶端。
* 每一次没有找到并且需要插入的时候。再插入一个记录后，把超出容量或者超出生存时间限制的记录全部删除。

## 3.6 ID

### 3.6.1 数据结构



ID.h

记录结构体含有六个字段：

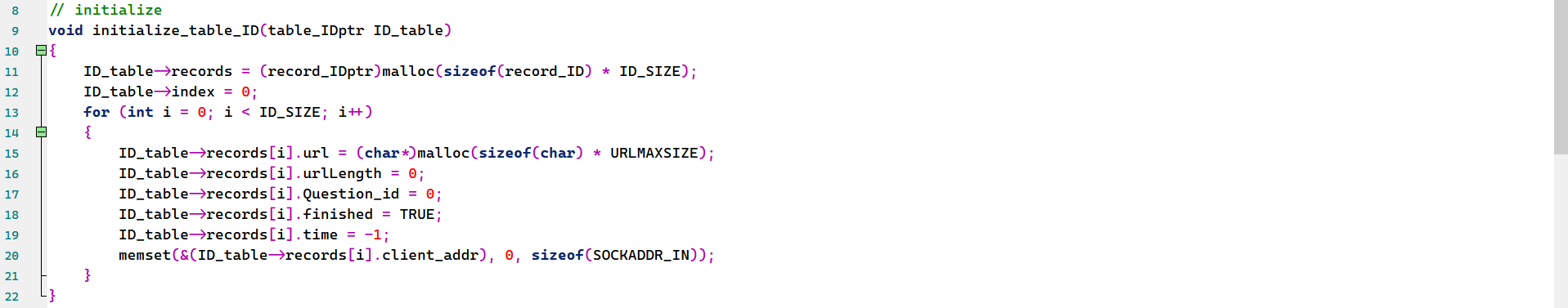
* url：域名指针
* urlLength：域名长度
* Question\_id：客户端发给DNS服务器的ID
* SOCKADDR\_IN client\_addr： 请求者的客户端套接字
* int time：进入的时间点（0-999）
* BOOL finished：状态标识，False标识该记录正在使用不可用，True标识可用

ID表结构体含有两个字段：

* record\_IDptr records：记录数组指针
* int index：当前查找空间所在的起始下标

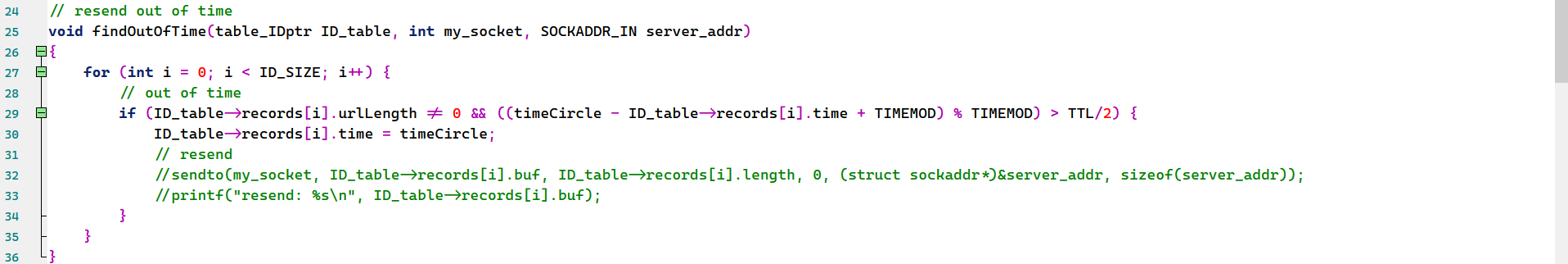
### 3.6.2 ID函数

**1. 创建ID表**



定量化数组空间申请，然后初始化index。

**2. 超时重传机制**



假如记录时间超过TTL一半，则触发重传。

根据PPT中的要求，该功能不予使用。

**3. 回应客户端报文时需要查找原的ID记录**



这里直接根据下标返回，速度极快。

4. **创建服务端报文时需要查找空间记录ID**



* 需求
  + 客户端发来的包，Cache和Tire字典树里未找到对应记录，转发给服务端进行DNS查询。
  + 针对多用户环境，不同地址发送来的ID可能相同，需要有独特的ID冲突避免机制。
  + 要保证性能：保证替换的效率；保证查找效率；保证空间利用率。
  + **存储ID&发送地址记录唯一标识**：针对多用户环境，不同地址发送来的ID可能相同，那么我们需要进行一个ID和发送地址的唯一标识来对应一个新的ID。这里的新ID定义为找到的ID表空间的下标+1。
  + **package的ID替换机制**：ID不进行大小端的转换，完全当作字节处理，使得给服务端的发送ID和接受ID字节一定相同，在本程序的识别也是相同的。目的是减小开销。
  + **ID表状态更新机制**：根据TTL的设定值进行查找状态更新。
* 具体机制

ID表状态更新机制：在每次查找到该记录空间时，更新状态，避免过多的刷新浪费性能。

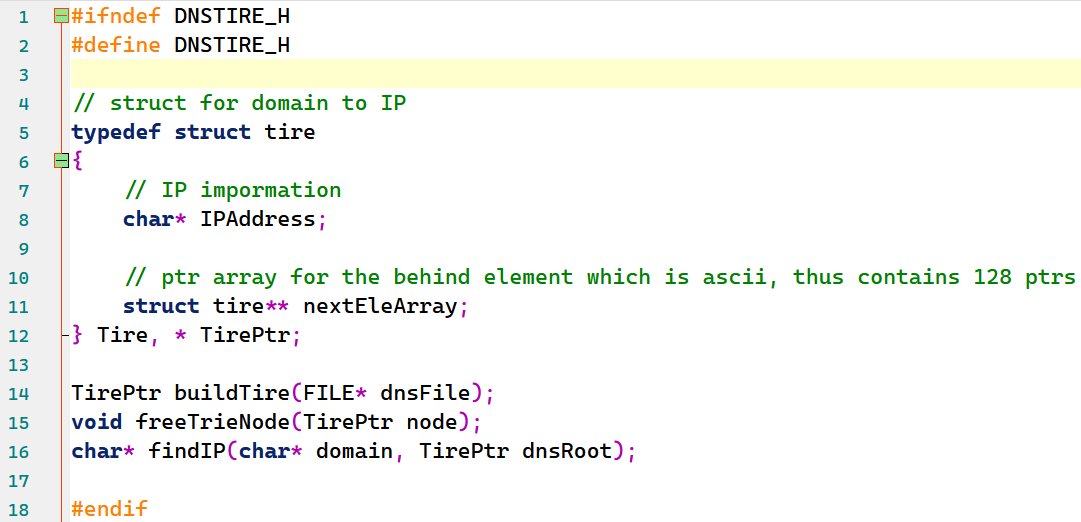
ID表寻找空间机制：初始化一个**index**，每次查找后让index为已经找到下标数值+1的新值，使得每次寻找都是可能性最大化。

## 3.7 dnsTire

### 3.7.1 数据结构

功能要求：根据本地文件读入信息，全字符匹配URL找出IP。

思路：因为本地数据量巨大，而且是全字符匹配类型，所以选择字典树进行存储。

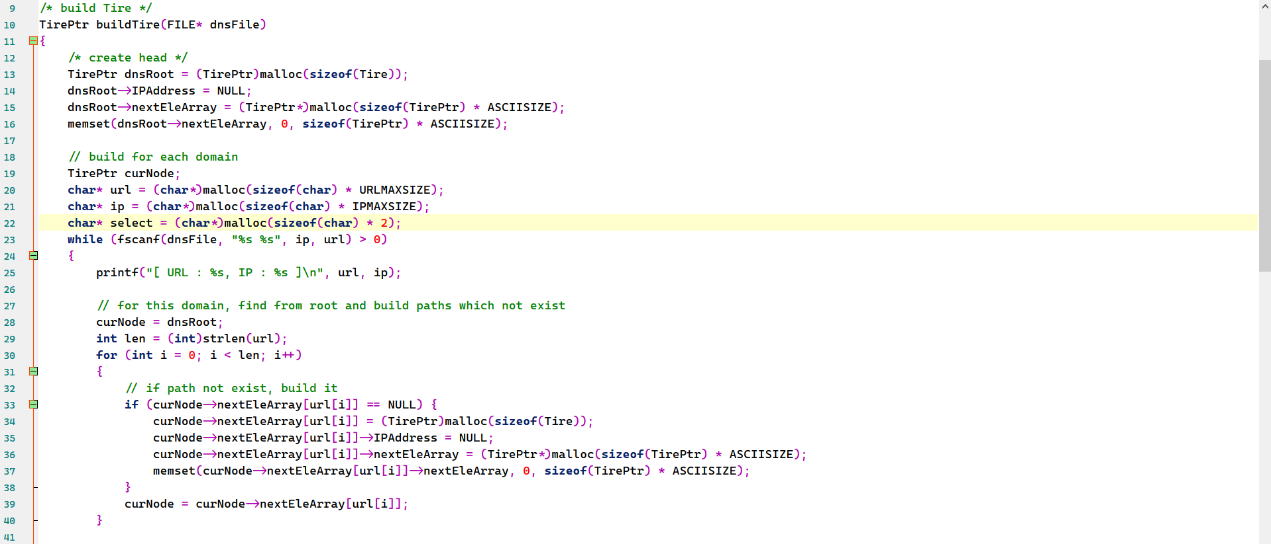


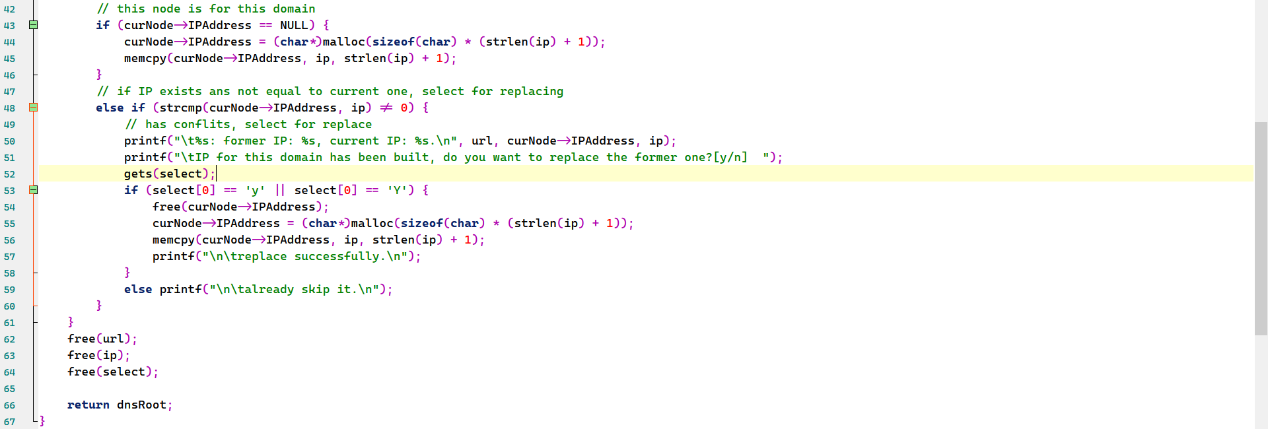
dnsTire.h

* 通过128（宏定义为ASCIISIZE）叉树来存字典树。
* ascii码直接转换为方向下标往下找，无需对应表。
* 通过牺牲空间来提高查询速度。域名多长找多少次。

### 3.7.2 dnsTire函数

**1. 建立字典树**





* 函数流程

1. 通过打开的文件进行读取，每一条记录一个循环

2. 从根节点开始逐步读入域名，下标代表字母。如果没有子树那么申请一下

3. 在域名读完的节点中存入IP信息

* 遇到的问题

1. 一个域名可能对应多个IP

解决方法：假如IP一样，直接跳过；不一样，根据用户输入查看是否替换。

2. 空间处理：没有的前缀节点就不申请空间

* 优点

1. 在数据量庞大的时候查找速度极快，查找复杂度只与查找数据长度有关。

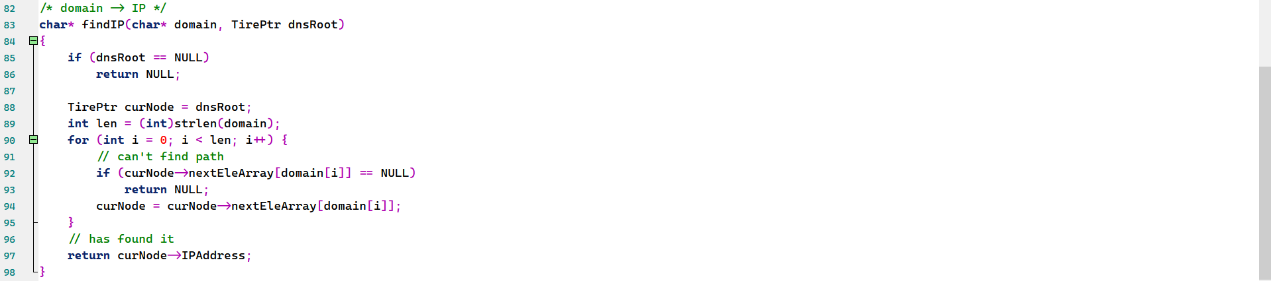
2. 空间处理使用malloc，减小栈的压力，提高性能。

**2. 释放字典树**



释放空间，进行内存回收管理。

**3. 根据URL查找对应IP**



非常简单的搜索机制。根据域名逐个搜索子树即可。

找到则返回IP指针；没找到则返回NULL。

## 3.8 主函数模块

### 3.8.1 准备阶段



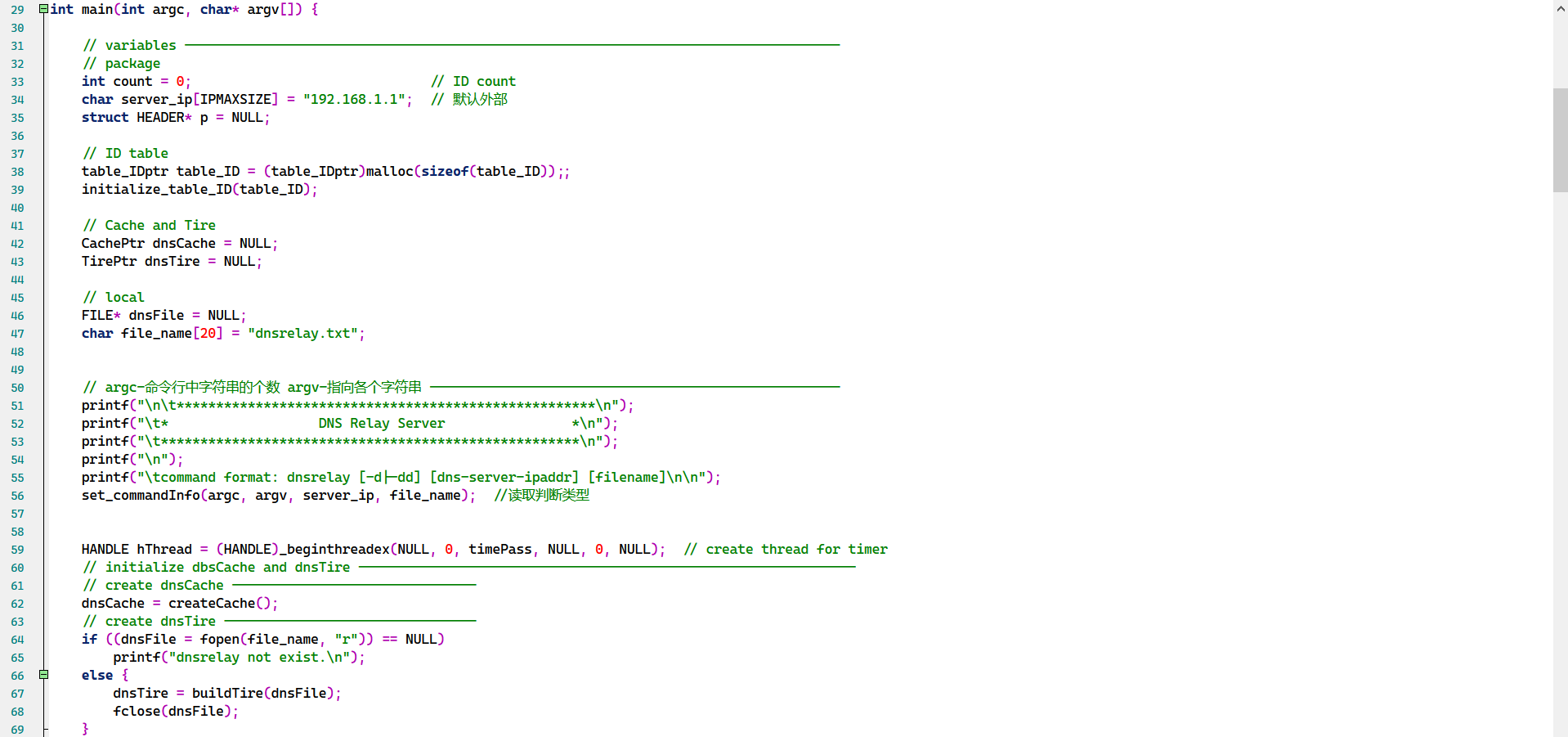
* 函数

1. 根据程序main输出初始化变量

2. 初始化套接字

* 全局变量

主要是套接字相关。



初始化各个变量。



初始化套接字。



工作时需要的局部变量。

### 3.8.2 工作阶段

对于每一次处理包，都是先以最少的步骤发送出去后再处理其他事情

鉴于代码过长，这里只讲伪代码和流程图:

while(1) {

收到一个包，存入buf

if (服务端来的) {

if (是屏蔽域名) 不予发送 // IP对于Ipv4/6的屏蔽机制

else {

在ID转换表里搜索对应的记录，转换ID后发送给对应客户端。// Ipv4/6都有

if (Ipv4) 将该记录加入Cache

}

}

// 从客户端来的

else {

if (Ipv4) {

cache里找

if (没找到) {

dnsTire（字典树）里找

if(找到了) 添加该记录到cache

}

if (cache或者tire找到了)

构造报文发给客户端 // 假如是屏蔽字段则会在构造报文时设置屏蔽报文

else {

使用ID表存放记录，构造报文发给服务端

if(ID表没找到空间存放) 扔掉  
 }

}

// Ipv6

else {

if (是屏蔽域名)continue;

使用ID表存放记录，构造报文发给服务端

}

}

}

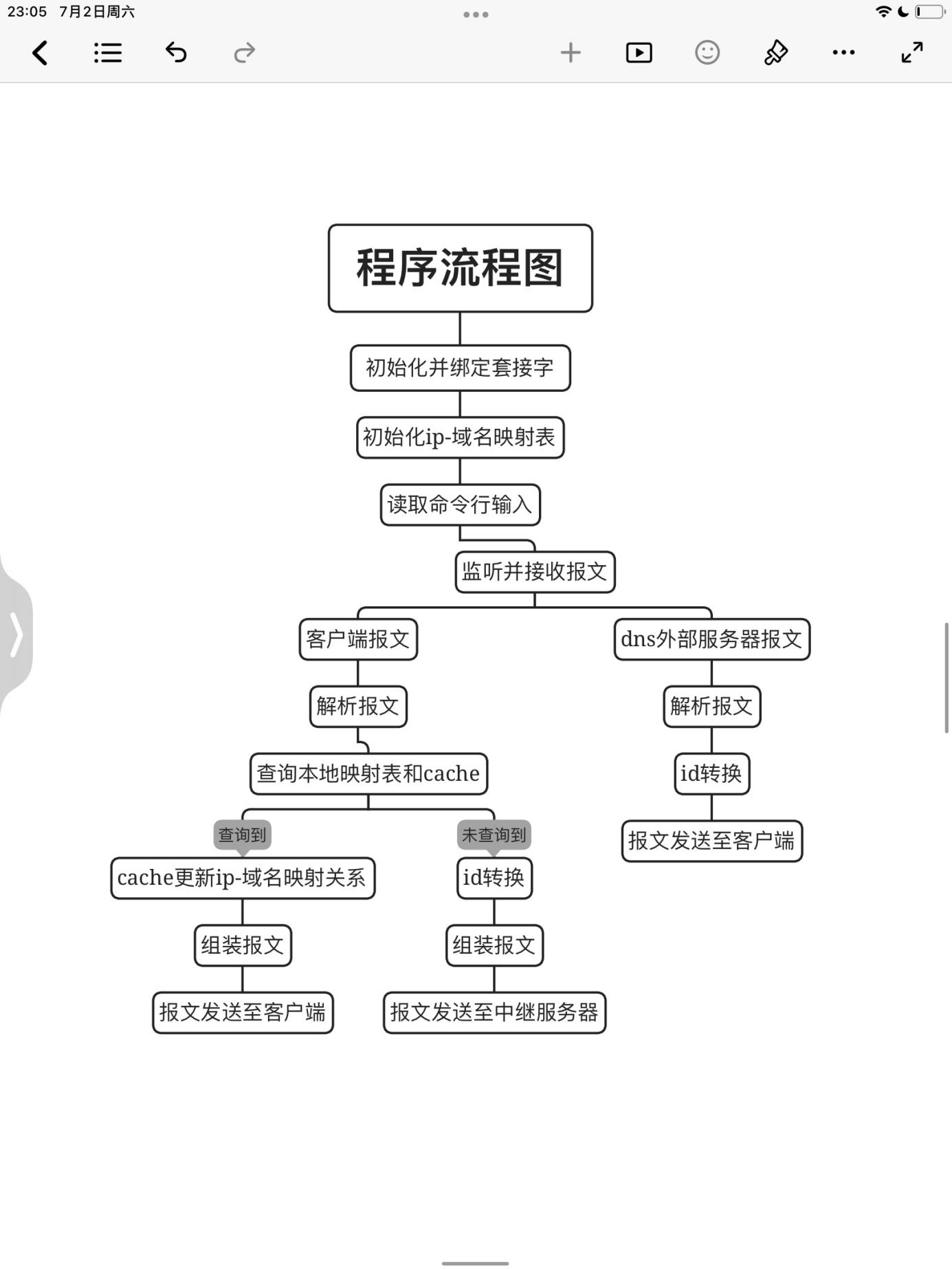
* **流程设计**

（1）从客户端获取报文，解析报文得到域名，在本地DNS和cache中查找报文中域名所对应的ip地址，若找到，构造并返回响应报文；

（2）从客户端获取报文，解析报文得到域名，在本地DNS和cache中查找报文中域名所对应的ip地址，未找到，存储信息至id转化表，修改报文id，转发请求报文至外部DNS服务器；

（3）从外部服务器获取报文，解析报文得到id，读取id转化表获得转发id，修改报文id转发至客户端；

* **流程图**



# 四：测试与调整

## 4.1 Cache机制调整

1. cache用普通单链表完全可以完成工作。所以并不采用所谓双向链表。

2. 在寻找时将找到的记录进行刷新，添加在记录最高处。

3. 在添加记录时，先在cache中寻找，找到则不用添加，查找时已经自动刷新位置。

4. 将删除超时和超出容量的步骤加在加入新记录时刻，避免过于频繁的删除，保证容量利用率。

## 4.2 Cache容量参数调整

在经过狂开浏览器新页面的测试环境下，最终生存时间设置为50，容量设置为20

此时，在一般使用浏览时也cache保持在95%的空间利用率。

## 4.3 ID表的机制调整

1. 模拟空间特性：（不涉及增删）使用数组方式，固定容量。

2. 搜索机制优化：设定index，固定下一次搜索的下标，使得开始搜索的位置一定是上次的下一个，使得搜索空间成功的概率最大化。

3. 使用空间下标+1作为转换后ID的唯一标识，在取出ID记录时通过下标取出，无需匹配。

4. TTL查询得知为10，所以设置TTL为10。在这个时间后不可能再次出现回应包

## 4.4 ID表的容量调整

在经过狂开浏览器新页面的测试环境下，最终TTL设置为10，容量设置为72。基本第一次即可找到可使用的ID转换空间。

## 4.5 各种宏定义参数的设置



对于轮回时间的处理：设为1000，基本不会出现1000s之内不上网的可能。



查询得知上述几个宏定义如上。

## 4.6 package处理问题

pachage主要处理和报文相关的内容，如读取报文中的域名，ip地址。

以及对报文内部信息的修改，如构造响应报文，请求报文等。

## 4.7 外部发送和Ipv6引出的屏蔽机制升级

测试[www.bilibili.com](http://www.bilibili.com)得知：Ipv4和Ipv6中的包可能都含有很多地址，所以必须两者都要实现屏蔽。

此外，外部攻击也有可能造成域名无法屏蔽，所以接收server端时也要进行屏蔽处理。

一开始仅针对Ipv6进行了外部屏蔽。没想到Ipv4外部攻击也需要进行屏蔽操作。

## 4.8 遇到的小问题

cache一开始设置的过小（10），导致替换删除极为频繁。

ID表设置过小（50）,经常丢包。

socket使用不当，不注重报文中存在\0字符，使得一开始出错。

# 五：正式测试结果及分析

## 5.1 本机测试



1. 主机发送dnsQ包到本地dns服务器

2. 先收到Ipv4的报文，在检测本地没有查找结果后，使用2号ID空间直接转发给服务器申请

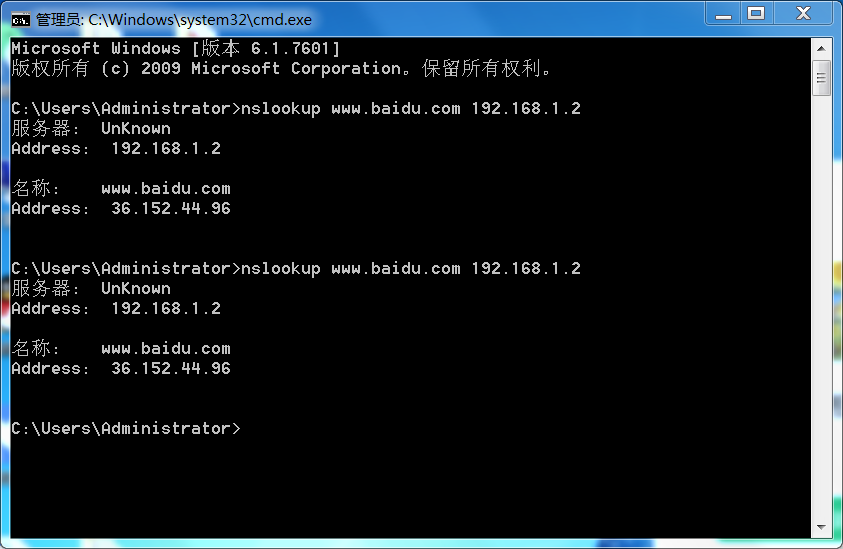
3. 收到Ipv4的R报文，将记录加入cache，并ID转换转发给客户端

4. 先收到Ipv6的报文，取出URL检测本地没有屏蔽为0.0.0.0后，使用1号ID空间转发给服务器

5. 收到Ipv6的R报文，我们对Ipv6并没有报文分析，无法将其加入cache，直接转发给客户端

## 5.2 外部主机测试



 win7系统

局域网内访问（前一个是上一次测试，不用管）

这里可以看到我们成功的转发了package。

## 5.3 屏蔽外部DNS服务器测试核心功能

首先将本机电脑的DNS服务器改为127.0.0.1

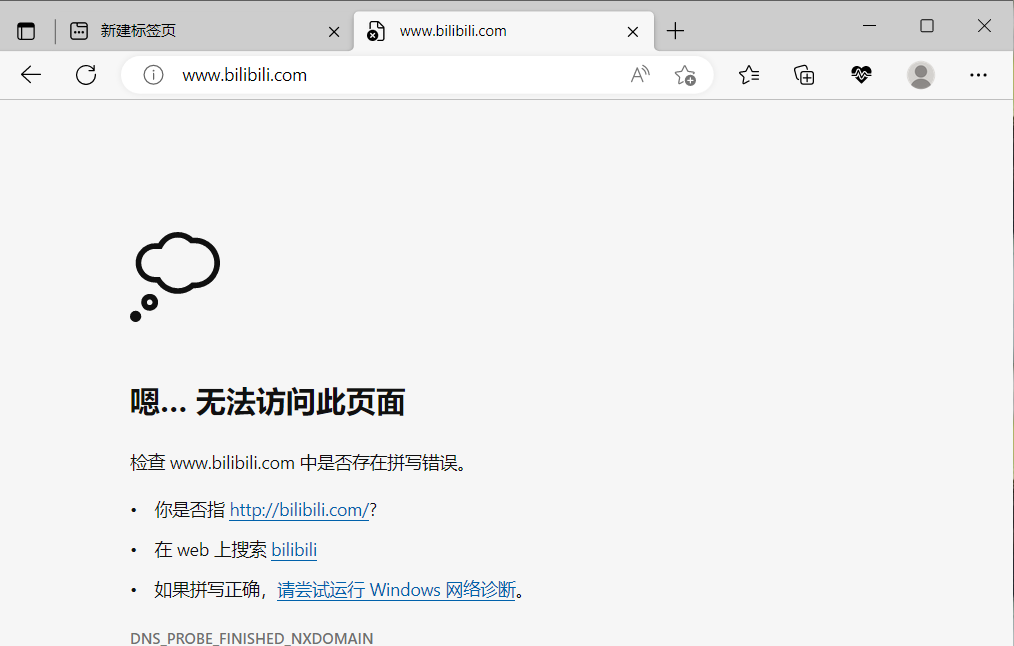


### 5.3.1 屏蔽功能

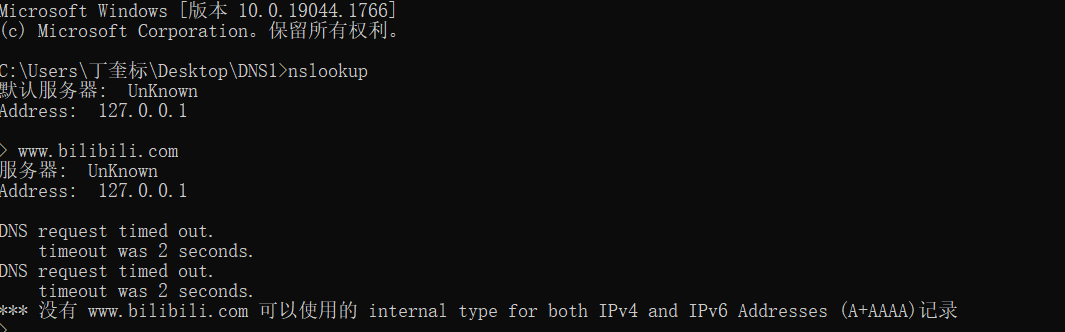
首先将www.bilibili.com加入屏蔽：（在文件里添加记录）



然后打开浏览器，输入网址www.bilibilib.com，可以看到无法刷新出网页



使用nslookp工具，可以看到如下，顺利完成屏蔽功能



* 解读

bilibili网站会发送dns的Ipv6和Ipv4包。我们都会进行屏蔽。

在第一个本机测试中可知，Ipv4和Ipv6中的包都含有很多地址，所以必须两者都要屏蔽。

### 5.3.2 本地查询功能

在本地文件中加入该条记录(其中IP地址为百度IP地址)：



在nslookup中进行查询如下：



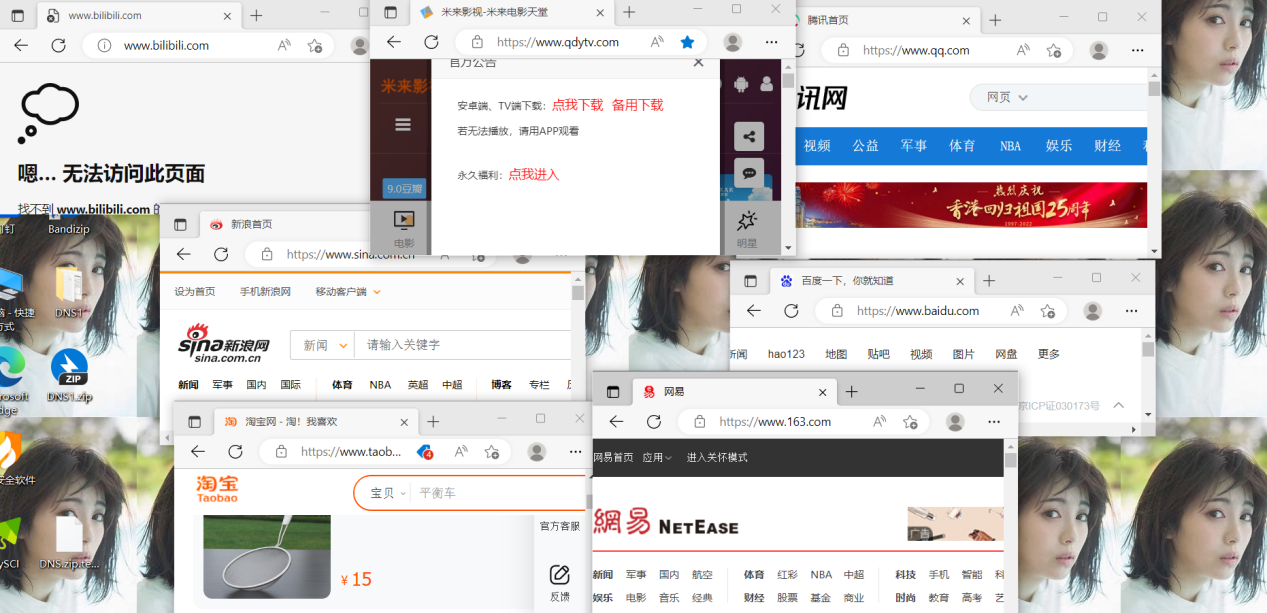
顺利返回对应IP地址。

### 5.3.3 外部查询功能&突发式查询测试

首先在浏览器中输入如下网址：

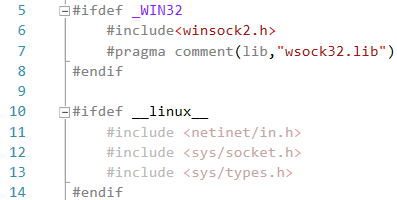


然后很快的依次点击回车，如下:



各个界面几乎同时刷新。

## 5.4 跨平台机制





最终我们在linux平台尝试了宏定义条件编译，发现很多变量类型并不通用。

## 5.5 字典接口

本地查询我们采用了**字典树**进行查找，长度即为查找次数，牺牲空间换取时间。

## 5.6 LRU缓冲池

我们对于**Cache**实现了LRU缓冲机制；

且对于宏参数定义进行了调试调整，使得最大程度上利用空间（频繁使用时在95%的利用率）

此外，在**ID**转换表中，我们也采用了类似的机制，是的每一次查找ID转换空间时都寻找的第一个一定是最大概率可用的空间。

# 六：实验总结

## 6.1 实验经历

本次实验我们花费了整整一周时间完成。

1. 一天时间将模块搭建好；
2. 两天时间进行模块编写和整合；
3. 三天时间针对各种样例和环境进行测试，优化代码
4. 最后花费一天时间撰写实验报告。

## 6.2 实验感想

在本次课程设计中，虽然是线上合作，小组成员还是通过腾讯会议等方式频繁进行沟通交流讨论。从课设初期的讨论程序模块的划分和明确各自的任务，再到代码编写过程中实现的细节以及遇到的问题，在沟通中一起理解一起进步。

无论是socket的应用还是报文的结构、含义，小组成员在本次实验之前对他们的了解都仅限于认识层次，所以在程序编写时进行的不断理解和结构梳理上花了较多的时间。在课设代码的编写过程中，理论知识在程序编写时的应用加深了小组成员对DNS服务器工作机制的理解。小组成员在程序初步完成后，为了提升程序响应的速度，决定再次对部分模块进行重新编写，包括了本地映射表以及cache的查询等。

小组成员在此次课设中合作编写DNS中继器，对于服务机制和过程的认识更加清晰，也磨练了团队合作的意识和能力。希望在以后的学习中，能够多多参与团队合作和讨论，有更好的状态学习专业知识。