

项目D结题报告

队名： 蛋黄酥

成员： 苏铃峰、邓淑仪、王泽双

报告类型： 结题报告

时间： 2015年9月20日

目录

[一、项目介绍及团队分工 3](#_Toc24546)

[二、项目执行中遇到的困难及解决方案 3](#_Toc1086)

[三、项目基本点完成情况 3](#_Toc17260)

[四、项目扩展点完成情况 3](#_Toc24269)

[五、项目心得及改进建议 3](#_Toc19212)

[六、源代码 3](#_Toc22943)

1. **项目****介绍及团队分工**

**项目D：**

1. **项目名称：**基于 IPv6 地址的复杂运算
2. **描述：**随着 IPv4 的枯竭，IPv6 马上面临大规模商用，地址长度从 IPv4 的 32 位变成了 128 位，地址数量庞大无比，过去的一些算法已不再适用。请在 Linux 下，用 C/C++ 设计一套数据结构、算法，满足 IPv4 和 IPv6 的基本运算。
3. **难度：**中
4. **总体要求及考核标准（总和100分）**
5. 基本点：

①、识别常见的 IPv6、IPv4 地址格式。例如：FE80:: 、FE80::/32 、210.34.0.0 、210.34.0.0/16。（10分）

②、从字典库中查询 IPv6、IPv4 单个地址、地址段的属性。字典库的格式（为地址段）：FE80::/32 属性1； FE81::/64 属性2； 210.34.0.0/16 属性3。（15分）

③、2 个字典库的差异比对。字典库的格式，同上。（15分）

④、2 个字典库做减，即字典库1减去字典库2中属性相同的地址段。字典库的格式，同上。（10分）

⑤、2 个字典库做加，即字典库1覆盖字典库2中属性不同的地址段，包括字典库2中不存在的地址段。字典库的格式，同上。（10分）

1. 拓展点：

①、附带详细设计文档。（10分）

②、合理的模块化编码。（15分）

③、良好的代码风格、质量。（15分）

**分工：**

姓名1：苏铃峰：队长，寻址模块，字典库对比模块，各模块整合

姓名2：邓淑仪：地址辨析模块

姓名3：王泽双：寻址模块、资料查找

1. 项目执行中遇到的困难及解决方案

困难1：ipv4和ipv6 地址解析时分别用什么方式

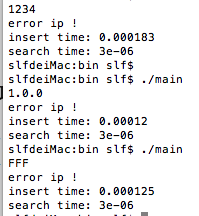
采取的解决方案：ipv4转化为数值总体大小判断，ipv6分8段判断

困难2：makefile编写

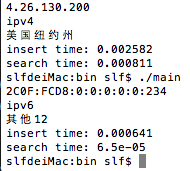
以前没有接触过，花了一些时间去学习，写了一个模版化的makefile

1. 项目基本点完成情况

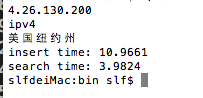
基本点1：



基本点2：

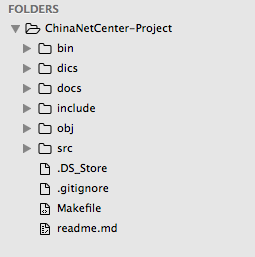


基本点3：582万条数据时的查询情况



1. 项目扩展点完成情况

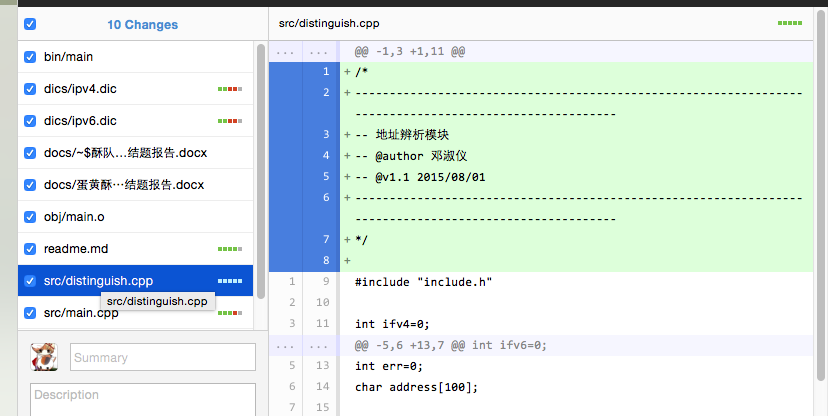
扩展点1：良好的模块化设计



扩展点2：良好的代码风格



扩展点3：使用git进行协同开发



1. 项目心得及改进建议

心得：

学习了ipv4和ipv6相关知识

对程序模块化设计有了更深的理解

对makefile的理解更加深入

学会使用git协同开发

1. 源代码

github地址：https://github.com/call-fold/ChinaNetCenter-Project

include.h

#ifndef INCLUDE\_H

#define INCLUDE\_H

#include <stdio.h>

#include <stdlib.h>

#include <iostream>

#include <vector>

#include <string>

#include <fstream>

#include <string.h>

using namespace std;

extern int ifv4;

extern int ifv6;

extern int err;

extern char address[100];

extern void if\_ipv4(char \*str);

extern void if\_ipv6(char \*str);

extern void distinguish(char \*str);

extern string temp\_str;

extern std::vector<string> svec1;

extern std::vector<string> svec2;

extern std::vector<string> svec3;

extern char ch[5];

extern int j;

extern int ipv4[4];

extern int ipv6\_min[8];

extern int ipv6\_max[8];

extern int ipv6\_input[8];

extern void clear\_ch();

extern int ipv4\_change(string stemp);

extern void ipv6\_change(string stemp,int \*x);

#endif

distinguish.cpp

/\*

-------------------------------------------------------------------------------------------------------

-- 地址辨析模块

-- @author 邓淑仪

-- @v1.1 2015/08/01

-------------------------------------------------------------------------------------------------------

\*/

#include "include.h"

int ifv4=0;

int ifv6=0;

int err=0;

char address[100];

//=======================================ipv4地址判断，元组解析

void if\_ipv4(char \*str)

{

int len;

int i;

int point\_count=0;

int num\_flage=0;

int num=0;

len=strlen(str);

for(i=0;i<len;i++)

{

if((str[i]>=48&&str[i]<=57)||str[i]==46||str[i]==58||(str[i]>=65&&str[i]<=70)||(str[i]>=97&&str[i]<=102))

{

if((str[i]>=48&&str[i]<=57)||str[i]==46)

{

if(str[i]==46&&num\_flage==1)

{

num=0;

point\_count++;

num\_flage=0;

}

else if(str[i]==46&&num\_flage==0)

{

err=1;

return;

}

else if(str[i]>=48&&str[i]<=57)

{

num\_flage=1;

num=num\*10+str[i]-48;

if(str[i+1]==46&&num>255)

{

err=1;

return;

}

}

}

else

{

ifv4=0;

return;

}

}

else

{

err=1;

return;

}

}

if(point\_count==3&&num<=255)

ifv4=1;

else

err=1;

return;

}

//=======================================ipv6地址判断，元组解析

void if\_ipv6(char \*str)

{

int len;

int i;

int num=0;

int colon\_count=0;

int doublecolon\_flage=0;

int colon\_flage=0;

len=strlen(str);

for(i=0;i<len;i++)

{

if((str[i]>=48&&str[i]<=57)||str[i]==46||str[i]==58||(str[i]>=65&&str[i]<=70)||(str[i]>=97&&str[i]<=102))

{

if((str[i]>=48&&str[i]<=57)||str[i]==58||(str[i]>=65&&str[i]<=70)||(str[i]>=97&&str[i]<=102))

{

if(str[i]==58)

{

if(colon\_flage==1&&doublecolon\_flage==0)

doublecolon\_flage=1;

else if(colon\_flage==1&&doublecolon\_flage==1)

{

err=1;

return;

}

if(colon\_flage==0)

{

num=0;

colon\_flage=1;

}

colon\_count++;

}

else

{

num++;

colon\_flage=0;

if(num>4)

{

err=1;

return;

}

}

}

else

{

ifv6=0;

return;

}

}

else

{

err=1;

return;

}

}

if(colon\_count<7&&colon\_count>=2&&doublecolon\_flage==1||colon\_count==7&&doublecolon\_flage==0)

ifv6=1;

else

err=1;

return;

}

void distinguish(char \*str)

{

if\_ipv4(str);

if(ifv4==0&&err==0)

if\_ipv6(str);

if(ifv4==0&&ifv6==0)

err=1;

return;

}

search.cpp

/\*

-------------------------------------------------------------------------------------------------------

-- 寻址模块

-- @author 苏铃峰

-- @v1.1 2015/09/18

-------------------------------------------------------------------------------------------------------

\*/

#include "include.h"

string temp\_str;

std::vector<string> svec1;

std::vector<string> svec2;

std::vector<string> svec3;

char ch[5];

int j=0;

int ipv4[4];

int ipv6\_min[8];

int ipv6\_max[8];

int ipv6\_input[8];

//===================================初始化

void clear\_ch(){

for (int i = 0; i < 5; ++i)

{

ch[i]=0;

}

j=0;

}

//============================================ipv4地址查找

int ipv4\_change(string stemp){

string::size\_type i=0;

int itemp=0;

vector<string>::size\_type sx=0;

for (int num = 0; num < 4; ++num){

for(;i!=stemp.size();++i){

if (stemp[i]=='.')

{

break;

}

ch[j++]=stemp[i];

}

ch[j]='\0';

ipv4[num]=atoi(ch);

clear\_ch();

++i;

}

itemp=ipv4[0]\*16777216+ipv4[1]\*65536+ipv4[2]\*256+ipv4[3];

return itemp;

}

//==============================================ipv6地址查找

void ipv6\_change(string stemp,int \*x){

string::size\_type i=0;

int itemp=0;

vector<string>::size\_type sx=0;

for (int num=0;num<8;++num){

for(;i!=stemp.size();++i){

if (stemp[i]==':')

{

break;

}

ch[j++]=stemp[i];

}

ch[j]='\0';

sscanf(ch,"%X",&x[num]);

clear\_ch();

++i;

}

}

main.cpp

/\*

-------------------------------------------------------------------------------------------------------

-- 主模块

-- @author 苏铃峰

-- @v1.1 2015/09/18

-------------------------------------------------------------------------------------------------------

\*/

#include "include.h"

#include "time.h"

int main(int argc, char const \*argv[])

{

clock\_t start, finish, middle;

double durationInsert, durationSearch;

start = clock();

string s\_ip;

int flag=0; // flag=1 false input,flag=2 ipv4,flag=3 ipv6

cin>>s\_ip;

for (int i = 0; i!=s\_ip.size(); ++i)

{

address[i]=s\_ip[i];

}

distinguish(address);

if(err==1)

flag=1;

else

{

if(ifv4==1&&ifv6==0)

flag=2;

if(ifv4==0&&ifv6==1)

flag=3;

}

if(flag==1){

cout<<"error ip !"<<endl;

}

else if(flag==2){

cout<<"ipv4"<<endl;

fstream outfile1("../dics/ipv4.dic",ios::out|ios::in); //输入ipv4字典库地址

while(outfile1>>temp\_str){

svec1.push\_back(temp\_str);

outfile1>>temp\_str;

svec2.push\_back(temp\_str);

outfile1>>temp\_str;

svec3.push\_back(temp\_str);

}

outfile1.close();

}

else if(flag==3){

cout<<"ipv6"<<endl;

fstream outfile2("../dics/ipv6.dic",ios::out|ios::in); //输入ipv6字典库地址

while(outfile2>>temp\_str){

svec1.push\_back(temp\_str);

outfile2>>temp\_str;

svec2.push\_back(temp\_str);

outfile2>>temp\_str;

svec3.push\_back(temp\_str);

}

outfile2.close();

}

middle = clock();

int ipv4\_min=0;

int ipv4\_max=0;

int ipv4\_input=0;

int flag\_indic=0; //==1 在字典中 ==0 不在字典中

vector<string>::size\_type sx=0;

for (int i = 0; i!=s\_ip.size(); ++i)

{

address[i]=s\_ip[i];

}

distinguish(address);

if(flag==2){

ipv4\_input=ipv4\_change(s\_ip);

for(;sx!=svec1.size();++sx){

ipv4\_min=ipv4\_change(svec1[sx]);

ipv4\_max=ipv4\_change(svec2[sx]);

if(ipv4\_input>=ipv4\_min&&ipv4\_input<=ipv4\_max){

flag\_indic=1;

break;

}

}

}

else if(flag==3){

ipv6\_change(s\_ip,ipv6\_input);

for(;sx!=svec1.size();++sx){

int ipv6\_flag=0;

ipv6\_change(svec1[sx],ipv6\_min);

ipv6\_change(svec2[sx],ipv6\_max);

for (int i = 0; i < 8; ++i)

{

if(ipv6\_min[i]<=ipv6\_input[i]&&ipv6\_input[i]<=ipv6\_max[i])

ipv6\_flag++;

}

if(ipv6\_flag==8){

flag\_indic=1;

break;

}

}

}

if(flag\_indic==1&&(flag==2||flag==3))

cout<<svec3[sx]<<endl;

else if(flag\_indic==0&&(flag==2||flag==3))

cout<<"ip is not in dic !"<<endl;

finish = clock();

durationInsert = (double)(middle - start)/CLOCKS\_PER\_SEC;

durationSearch = (double)(finish - middle)/CLOCKS\_PER\_SEC;

cout<<"insert time: "<<durationInsert<<endl; //插入时间

cout<<"search time: "<<durationSearch<<endl; //搜索时间

return 0;

}

makefile

DIR\_INC = ./include

DIR\_SRC = ./src

DIR\_OBJ = ./obj

DIR\_BIN = ./bin

SRC = $(wildcard ${DIR\_SRC}/\*.cpp)

OBJ = $(patsubst %.cpp,${DIR\_OBJ}/%.o,$(notdir ${SRC}))

TARGET = main

BIN\_TARGET = ${DIR\_BIN}/${TARGET}

CC = g++

CFLAGS = -g -Wall -I${DIR\_INC}

${BIN\_TARGET}:${OBJ}

$(CC) $(OBJ) -o $@

${DIR\_OBJ}/%.o:${DIR\_SRC}/%.cpp

$(CC) $(CFLAGS) -c $< -o $@

.PHONY:clean

clean:

find ${DIR\_OBJ} -name \*.o -exec rm -rf {} \;