

## **Link Addressing**

|  |  |
| --- | --- |
| Name: | Wilson Vidyut Doloy |
| Reg No: | 19BCE1603 |
| Subject: | CSE1004 Network and Communication Lab |
| Slot: | L52+L53 (Prof. Ganesan) |
| Date: | 16 September 2020 |

Check Validity of Network address

Find its class

Find its default net mask

Find network Address(first id)

Find Host ID

Find Number of hosts network can hold

Server Side Code:

#include<stdio.h>  
#include<netinet/in.h>  
#include<sys/types.h>  
#include<sys/socket.h>  
#include<netdb.h>  
#include<string.h>  
#include<stdlib.h>  
#define MAX 20  
#define PORT 43454  
#define SA struct sockaddr  
void func(int sockfd)  
{  
char buff[MAX];  
int ipclass, validity[10], dot[10], k=0, g=0, c=0, h=0, check=0, range, t, i;  
char ipcs[20]="", ip[MAX], ipcopy[40]="", class[1], hostID[30]="",hosts[20]="", netID[20]="", ntfo[10]="", defaultmask[20];  
int n,clen;  
struct sockaddr\_in cli;  
clen=sizeof(cli);  
  
  
bzero(ip,MAX);  
recvfrom(sockfd,ip,sizeof(ip),0,(SA \*)&cli,&clen);  
  
  
strcpy(ipcopy,ip);  
char \* token = strtok(ip, ".");  
   
 // loop through the string to extract all other tokens  
 while(token != NULL)   
 {  
 sscanf(token,"%d",&t);  
 validity[g++]=t;//storing each token  
 token = strtok(NULL, ".");  
 }  
   
 //IP VALIDATION  
 if(g==4)  
 {  
 for(i=0;i<4;i++)  
 {  
 if(validity[i]>=0 && validity[i]<=255)  
 check++;  
 else  
 {  
 check=0;  
 break;  
 }  
 }  
 }  
 else  
 {  
 check=0;  
 }  
  
 if(check==4)  
 {  
 //printf("VALID IP\n");  
 char val[MAX]="VALID IP";  
 sendto(sockfd,val,sizeof(val),0,(SA \*)&cli,sizeof(cli));  
 bzero(val,MAX);  
   
 for(i=0;i<strlen(ipcopy);i++)  
 {  
 if(ipcopy[i]=='.')  
 {  
 dot[k++]=i;  
 }  
 }  
  
   
 //CLASS, MASK, ADDRESS, HOSTID, No OF HOSTS CALCULATION  
 ipclass=validity[0];  
 if(ipclass<128)  
 {  
 class[0]='A';  
 strcpy(defaultmask,"255.0.0.0");  
 h=dot[0]+1;  
 strcpy(ntfo,"0.0.0");  
 strcpy(hosts,"16777214");  
   
 }  
 if(ipclass>127 && ipclass<192)  
 {  
 class[0]='B';  
 strcpy(defaultmask,"255.255.0.0");  
 h=dot[1]+1;  
 strcpy(ntfo,"0.0");  
 strcpy(hosts,"65534");  
   
 }  
 if(ipclass>191 && ipclass<224)  
 {  
 class[0]='C';  
 strcpy(defaultmask,"255.255.255.0");  
 h=dot[2]+1;  
 strcpy(ntfo,"0");  
 strcpy(hosts,"254");  
   
 }  
 if(ipclass>223 && ipclass<240)  
 {  
 class[0]='D';  
 strcpy(defaultmask,"not defined");  
 strcpy(hosts,"not defined");  
 h=-1;  
 }  
 if(ipclass>239 && ipclass<256)  
 {  
 class[0]='E';  
 strcpy(defaultmask,"not defined");  
 strcpy(hosts,"not defined");  
 h=-1;  
 }   
   
 sendto(sockfd,class,sizeof(class),0,(SA \*)&cli,sizeof(cli));  
 bzero(class,1);  
  
 sendto(sockfd,defaultmask,sizeof(defaultmask),0,(SA \*)&cli,sizeof(cli));  
 bzero(defaultmask,20);  
   
 if(h==-1)  
 { char a[40] = "Starting Address: not defined\n";  
 char b[30] = "HostID: not defined\n";  
   
 sendto(sockfd,a,sizeof(a),0,(SA \*)&cli,sizeof(cli));  
 bzero(a,40);  
   
 sendto(sockfd,b,sizeof(b),0,(SA \*)&cli,sizeof(cli));  
 bzero(b,30);  
 sendto(sockfd,hosts,sizeof(hosts),0,(SA \*)&cli,sizeof(cli));  
 bzero(hosts,20);  
 }  
 else  
 {  
 memset(netID, '\0', sizeof(netID));  
 strncpy(netID, ipcopy, h);  
 sendto(sockfd,netID,sizeof(netID),0,(SA \*)&cli,sizeof(cli));  
 bzero(netID,20);  
 sendto(sockfd,ntfo,sizeof(ntfo),0,(SA \*)&cli,sizeof(cli));  
 bzero(ntfo,10);  
 while (c<strlen(ipcopy))   
 {  
 hostID[c] = ipcopy[h+c];  
 c++;  
 }  
 hostID[c] = '\0';  
 sendto(sockfd,hostID,sizeof(hostID),0,(SA \*)&cli,sizeof(cli));  
 bzero(hostID,30);  
 sendto(sockfd,hosts,sizeof(hosts),0,(SA \*)&cli,sizeof(cli));  
 bzero(hosts,20);  
 }  
 }  
 else{  
 char val[MAX]="INVALID IP";  
 sendto(sockfd,val,sizeof(val),0,(SA \*)&cli,sizeof(cli));  
 bzero(val,MAX);  
 //printf("INVALID IP\n");  
 }  
  
  
}  
int main()  
{  
int sockfd;  
struct sockaddr\_in servaddr;  
sockfd=socket(AF\_INET,SOCK\_DGRAM,0);  
if(sockfd==-1)  
{  
printf("socket creation failed...\n");  
exit(0);  
}  
else  
printf("Socket successfully created..\n");  
bzero(&servaddr,sizeof(servaddr));  
servaddr.sin\_family=AF\_INET;  
servaddr.sin\_addr.s\_addr=htonl(INADDR\_ANY);  
servaddr.sin\_port=htons(PORT);  
if((bind(sockfd,(SA \*)&servaddr,sizeof(servaddr)))!=0)  
{  
printf("socket bind failed...\n");  
exit(0);  
}  
else  
printf("Socket successfully binded..\n");  
func(sockfd);  
close(sockfd);  
}

Client Side Code:

#include<sys/socket.h>  
#include<netdb.h>  
#include<string.h>  
#include<stdlib.h>  
#include<stdio.h>  
#include <sys/types.h>  
#include <netinet/in.h>  
#define MAX 20  
#define PORT 43454  
#define SA struct sockaddr  
int main()  
{  
char buff[MAX];  
int ipclass, validity[10], dot[10], k=0, g=0, c=0, h=0, check=0, range, t, i;  
char ipcs[20]="", ip[MAX], ipcopy[40]="", class[1], hosts[20]="", hostID[30]="", netID[20]="", ntfo[10]="", defaultmask[20];  
  
int sockfd,len,n,r;  
struct sockaddr\_in servaddr,cli;  
socklen\_t servlen;  
  
sockfd=socket(AF\_INET,SOCK\_DGRAM,0);  
if(sockfd==-1)  
{  
printf("socket creation failed...\n");  
exit(0);  
}  
else  
printf("Socket successfully created..\n");  
bzero(&servaddr,sizeof(len));  
servaddr.sin\_family=AF\_INET;  
servaddr.sin\_addr.s\_addr=inet\_addr("127.0.0.1");  
servaddr.sin\_port=htons(PORT);  
  
  
len=sizeof(servaddr);  
  
  
printf("Enter an IP Address: ");  
scanf("%s",ip);  
sendto(sockfd,ip,sizeof(ip),0,(SA \*)&servaddr,len);  
  
  
char val[MAX];  
bzero(val,MAX);  
recvfrom(sockfd,val,sizeof(val),0,(SA \*)&servaddr,&len);  
printf("%s\n",val);  
  
bzero(class,1);  
recvfrom(sockfd,class,sizeof(class),0,(SA \*)&servaddr,&len);  
printf("Class of %s is: %s \n",ip,class);  
bzero(ip,sizeof(ip));  
  
bzero(defaultmask,20);  
recvfrom(sockfd,defaultmask,sizeof(defaultmask),0,(SA \*)&servaddr,&len);  
printf("Default Mask: %s\n",defaultmask);  
  
if(class[0]=='D' || class[0]=='E')  
{  
char a[40]="",b[30]="";  
bzero(a,40);  
recvfrom(sockfd,a,sizeof(a),0,(SA \*)&servaddr,&len);  
printf("%s",a);  
bzero(b,30);  
recvfrom(sockfd,b,sizeof(b),0,(SA \*)&servaddr,&len);  
printf("%s",b);  
bzero(hosts,20);  
recvfrom(sockfd,hosts,sizeof(hosts),0,(SA \*)&servaddr,&len);  
printf("Number of hosts: %s\n",hosts);  
}  
else if(class[0]=='A' || class[0]=='B' || class[0]=='C')  
{  
bzero(netID,20);  
recvfrom(sockfd,netID,sizeof(netID),0,(SA \*)&servaddr,&len);  
bzero(ntfo,10);  
recvfrom(sockfd,ntfo,sizeof(ntfo),0,(SA \*)&servaddr,&len);  
printf("Starting Address: %s%s\n",netID,ntfo);  
  
bzero(hostID,30);  
recvfrom(sockfd,hostID,sizeof(hostID),0,(SA \*)&servaddr,&len);  
printf("Host ID: %s\n",hostID);  
  
bzero(hosts,20);  
recvfrom(sockfd,hosts,sizeof(hosts),0,(SA \*)&servaddr,&len);  
printf("Number of Hosts: %s\n",hosts);  
}  
else  
close(sockfd);  
}

OUTPUT:



















