**Name: Abinash Satapathy**

**Reg. No.: 16BCE0081**

**Slot: L27 + L28**

**Subject: Networking Lab (CSE1004)**

**Experiment – 4**

1. Classful/less addressing

#include<stdio.h>

#include<string.h>

// Function to find out the Class

char findClass(char str[])

{

// storing first octet in arr[] variable

char arr[4];

int i = 0;

while (str[i] != '.')

{

arr[i] = str[i];

i++;

}

i--;

// converting str[] variable into number for

// comparison

int ip = 0, j = 1;

while (i >= 0)

{

ip = ip + (str[i] - '0') \* j;

j = j \* 10;

i--;

}

// Class A

if (ip >=1 && ip <= 126)

return 'A';

// Class B

else if (ip >= 128 && ip <= 191)

return 'B';

// Class C

else if (ip >= 192 && ip <= 223)

return 'C';

// Class D

else if (ip >= 224 && ip <= 239)

return 'D';

// Class E

else

return 'E';

}

// Function to separate Network ID as well as

// Host ID and print them

void separate(char str[], char ipClass)

{

// Initializing network and host array to NULL

char network[12], host[12];

for (int k = 0; k < 12; k++)

network[k] = host[k] = '\0';

// for class A, only first octet is Network ID

// and rest are Host ID

if (ipClass == 'A')

{

int i = 0, j = 0;

while (str[j] != '.')

network[i++] = str[j++];

i = 0;

j++;

while (str[j] != '\0')

host[i++] = str[j++];

printf("Network ID is %s\n", network);

printf("Host ID is %s\n", host);

}

// for class B, first two octet are Network ID

// and rest are Host ID

else if (ipClass == 'B')

{

int i = 0, j = 0, dotCount = 0;

// storing in network[] up to 2nd dot

// dotCount keeps track of number of

// dots or octets passed

while (dotCount < 2)

{

network[i++] = str[j++];

if (str[j] == '.')

dotCount++;

}

i = 0;

j++;

while (str[j] != '\0')

host[i++] = str[j++];

printf("Network ID is %s\n", network);

printf("Host ID is %s\n", host);

}

// for class C, first three octet are Network ID

// and rest are Host ID

else if (ipClass == 'C')

{

int i = 0, j = 0, dotCount = 0;

// storing in network[] up to 3rd dot

// dotCount keeps track of number of

// dots or octets passed

while (dotCount < 3)

{

network[i++] = str[j++];

if (str[j] == '.')

dotCount++;

}

i = 0;

j++;

while (str[j] != '\0')

host[i++] = str[j++];

printf("Network ID is %s\n", network);

printf("Host ID is %s\n", host);

}

// Class D and E are not divided in Network

// and Host ID

else

printf("In this Class, IP address is not"

" divided into Network and Host ID\n");

}

// Driver function is to test above function

int main()

{

char str[] = "192.226.12.11";

char ipClass = findClass(str);

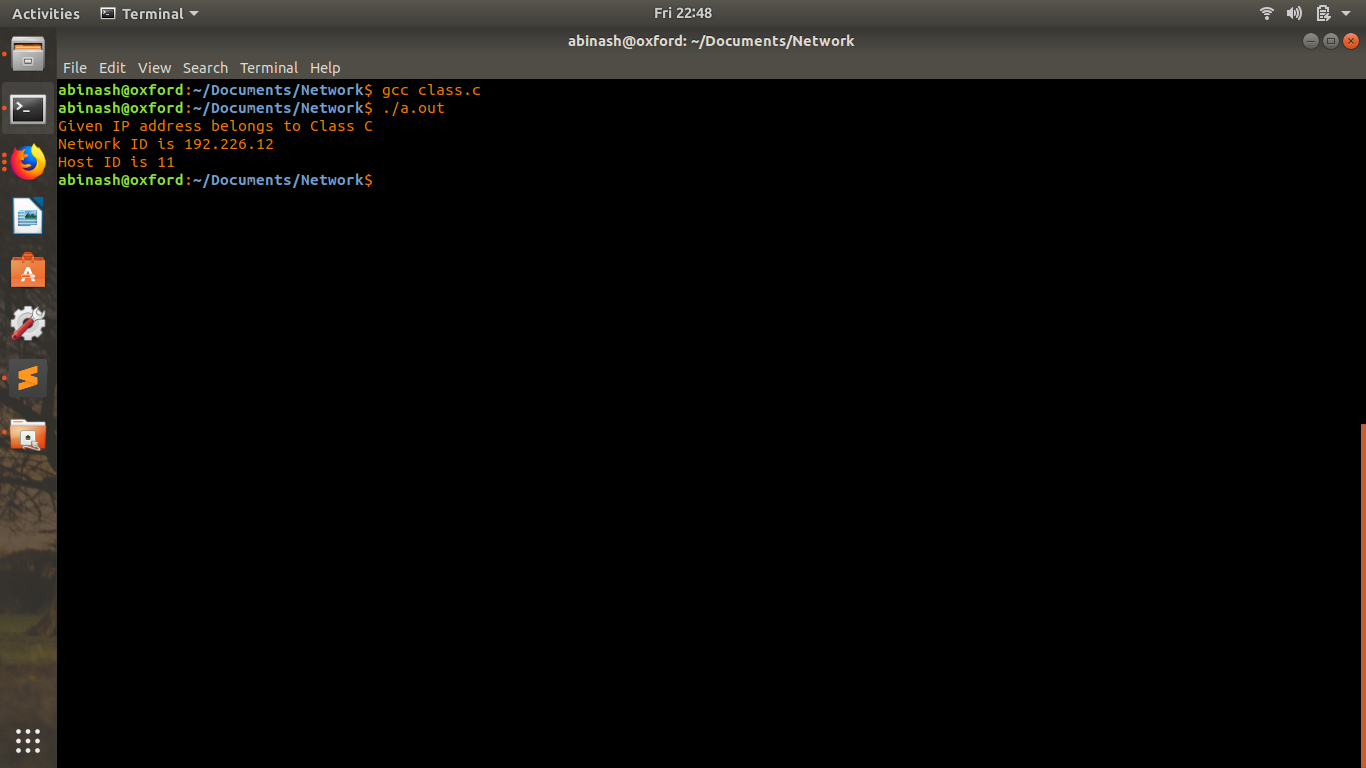
printf("Given IP address belongs to Class %c\n",

ipClass);

separate(str, ipClass);

return 0;

}



1. Subnetting

#include <iostream>

using namespace std;

#include <stdio.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <netinet/in.h>

#include <errno.h>

#include <string.h>

#include <stdlib.h>

void setIPv4(char \*ip,char \*gw,char \*netmask)

{

char cmd[128];

//network interface

char nwkInf[5]="eth0";

//link down command in Linux

sprintf(cmd,"ip link set %s down",nwkInf);

system(cmd);

memset(cmd,0x00,64);

//command to set ip address, netmask

sprintf(cmd,"ifconfig %s %s netmask %s",nwkInf,ip,netmask);

system(cmd);

printf("\ncmd : %s",cmd); fflush(stdout);

memset(cmd,0X00,64);

//command to set gateway

sprintf(cmd,"route add default gw %s %s",gw,nwkInf);

system(cmd);

memset(cmd,0X00,64);

//link up command

sprintf(cmd,"ip link set %s up",nwkInf);

system(cmd);

}

int main()

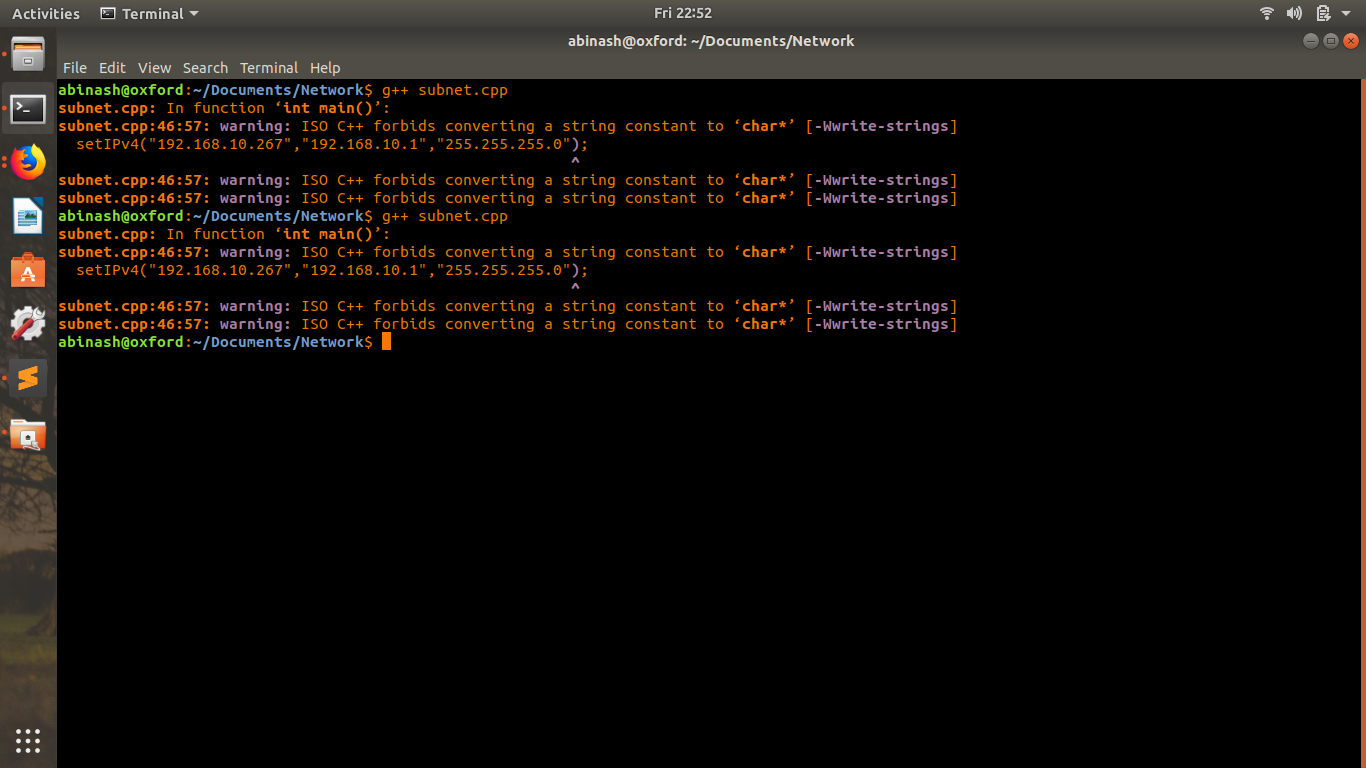
{

//calling function to set network settings

setIPv4("192.168.10.267","192.168.10.1","255.255.255.0");

return 0;

}



1. Distance Vector

#include <iostream>

#include <stdio.h>

using namespace std;

struct node {

int dist[20];

int from[20];

} route[10];

int main()

{

int dm[20][20], no;

cout << "Enter no of nodes." << endl;

cin >> no;

cout << "Enter the distance matrix:" << endl;

for (int i = 0; i < no; i++) {

for (int j = 0; j < no; j++) {

cin >> dm[i][j];

/\* Set distance from i to i as 0 \*/

dm[i][i] = 0;

route[i].dist[j] = dm[i][j];

route[i].from[j] = j;

}

}

int flag;

do {

flag = 0;

for (int i = 0; i < no; i++) {

for (int j = 0; j < no; j++) {

for (int k = 0; k < no; k++) {

if ((route[i].dist[j]) > (route[i].dist[k] + route[k].dist[j])) {

route[i].dist[j] = route[i].dist[k] + route[k].dist[j];

route[i].from[j] = k;

flag = 1;

}

}

}

}

} while (flag);

for (int i = 0; i < no; i++) {

cout << "Router info for router: " << i + 1 << endl;

cout << "Dest\tNext Hop\tDist" << endl;

for (int j = 0; j < no; j++)

printf("%d\t%d\t\t%d\n", j+1, route[i].from[j]+1, route[i].dist[j]);

}

return 0;

}

