```
//Server
#include<stdio.h>
#include<stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include<netinet/in.h>// Used for network and node byte ordering
#include <errno.h>// this is useful if error is present then perror will be triggered from our code
#include<string.h>
#include<time.h>
#include<unistd.h>
#include<pthread.h>

#define MAX 1024


/*
First let us recall that we need to create a  socket
THen bind
THen we need to recievefrom in case of UDP
Then we need to sendfrom in case of UDP
*/
/*
```

sockaddr_in has the following predefined

sin_family -> denoting which domain we will be using

sin_port -> the port number

sin_addr (of type in_addr) -> IP address

sin_zero of 8 bits just to pad

whereas

sockaddr is the one which is accepted by the API Calls

sa_family -> denoting to which domain it will belong to

sa_data -> it is the socket(port+IP address)

in_addr is another data structure having

s_addr -> where IP address is entered

*/

int sock, cli;// these are server and client descriptors they will be created when we etsablish sockets

// DEscriptor is a simple file descriptor in UNIX

int len;// to get length of sockaddr_in structure

```
struct sockaddr_in server, client; // declaring sockaddr_in structure variable as server and client by which we will be connecting
int count = 0;
struct table{
   int i ;
   struct sockaddr_in client;
};

struct table arr[10];// creating by default 10 clients at max



int main(){

   // First parameter that is provided to the socket is the domain like IPv4, IPv6 we can provide
   // SOCK_DGRAM says that we are send over UDP the packets
   // We have set the protocol field to

   if((sock = socket(AF_INET,SOCK_DGRAM,0))==-1){
      // THis provides error checking and we will exit if socket is not created
      perror("Socket");
      exit(-1);
   }
```

```
   //NOw let us start to bind step as we have created a socket
   server.sin_family = AF_INET;// THis denotes the family of the server IPv4
   server.sin_port = htons(5000);// THis is used for host byte order to network byte order
   server.sin_addr.s_addr = INADDR_ANY;// THis says the server to bind to any of the client present on this local system
   bzero(&server.sin_zero,8);// this is filled with 8 zeros by bzero function


   len = sizeof(struct sockaddr_in);


   // Binding will be done now


   if ((bind(sock,(struct sockaddr*)&server,len))==-1){
      //Error checking in binding
      perror("Bind");
      exit(-1);
   }



   // Now we have entered recievefrom phase of the server
```

```
   while(1){
      char buffer [MAX];
      int n = recvfrom(sock,(char*)buffer,MAX,MSG_WAITALL,(struct
sockaddr*)&client,&len);
      printf("Message recieved! from %d\n",client.sin_port);


      // recvfrom will accept socket of server which is created
      // pointer to the buffer where dtat from client will be filled
      // flag to it will be taken
      // also it returns a client sockaddr values
      // It takes the maximum segment size that it can accept
      if (buffer[0] == 'a'){
         printf("Came in \n");
         arr[count].i = buffer[1] - 'a';
         arr[count].client = client;
         count++;
         //sending port number will be enough in this case as in local
machine all have same IP
         // send all table values to the client connected newly so that
he will update
         for(int i=0;i<count;i++){
            char port [10];
```

```
            int port_num = arr[i].client.sin_port;
            snprintf(port,10,"%d",port_num);
            printf("THe client will be sent all the values in the table
%d\n",port_num);
            sendto(sock,(char*)port,strlen(port),MSG_CONFIRM,(struct
sockaddr*) &client,len);
        }
        char buffer []= "END";

sendto(sock,(char*)buffer,strlen(buffer),MSG_CONFIRM,(struct
sockaddr*) &client,len);
        for(int i=0;i<count;i++){
            char port [10];
            int port_num = client.sin_port;
            snprintf(port,10,"%d",port_num);
            sendto(sock,(char*)port, strlen(port),MSG_CONFIRM,(struct
sockaddr*)&arr[i].client,len);

sendto(sock,(char*)buffer,strlen(buffer),MSG_CONFIRM,(struct
sockaddr*) &client,len);


        }
      }


  }
  return 0;
```

```
}
```

```
// Client A
#include<stdio.h>
#include<stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include<netinet/in.h>// Used for network and node byte ordering
#include <errno.h>// this is useful if error is present then perror will be triggered from our code
#include<string.h>
#include<time.h>
#include<unistd.h>

#include<pthread.h>

#define MAX 1024


/*
First let us recall that we need to create a  socket
THen bind
THen we need to recievefrom in case of UDP
```

Then we need to sendfrom in case of UDP

*/

/*

sockaddr_in has the following predefined

sin_family -> denoting which domain we will be using

sin_port -> the port number

sin_addr (of type in_addr) -> IP address

sin_zero of 8 bits just to pad

whereas

sockaddr is the one which is accepted by the API Calls

sa_family -> denoting to which domain it will belong to

sa_data -> it is the socket(port+IP address)

in_addr is another data structure having

s_addr -> where IP address is entered

*/

```
int sock, cli;// these are server and client descriptors they will be created when we etsablish sockets

int len = sizeof(struct sockaddr_in);// to get length of sockaddr_in structure

struct sockaddr_in server, client,myself,temp,seen,p; // declaring sockaddr_in structure variable as server and client by which we will be connecting

int count = 0;

struct table{
   int i ;
   struct sockaddr_in client;
};


char buffer [MAX];


int n= 0;


char recBuff [MAX];


struct table arr[10];// creating by default 10 clients at max
```

```
void* sendToall(void* args){
   printf("Enter the message to send\n");
   while((buffer[n++]=getchar())!='\n');
   for(int i =0;i<count;i++){
      sendto(sock,(char*)buffer,strlen(buffer),MSG_CONFIRM,(struct
sockaddr*)&arr[i].client,len);
   }

}

void* recServer(void *args){
   recvfrom(sock,(char*)recBuff,MAX,MSG_WAITALL,(struct
sockaddr*)&client,&len);
   if (client.sin_port == server.sin_port && seen.sin_port!=
client.sin_port){
      printf("Message recieved from server %d\n",atoi(recBuff));
      arr[count].i = count;
      temp.sin_family = AF_INET;// THis denotes the family of the
server IPv4
      temp.sin_port = atoi(recBuff);// THis is used for host byte order
to network byte order
      temp.sin_addr.s_addr = INADDR_ANY;// THis says the server to
bind to any of the client present on this local system
      bzero(&temp.sin_zero,8);
```

```
        arr[count].client = temp;
        count++;
        seen = client;
    }
    else if(seen.sin_port!=client.sin_port){
        printf("Message recieved from, %d :
%s\n",client.sin_port,recBuff);
    }
    else{
        return;
    }
    sleep(5);

}
```

```
int main(){
    // WE will be creating a client with bind so that we won't have any
confusion

    if((sock = socket(AF_INET,SOCK_DGRAM,0))==-1){
        // THis provides error checking and we will exit if socket is not
created
```

```
    perror("Socket");
    exit(-1);
  }


  server.sin_family = AF_INET;// THis denotes the family of the server IPv4
  server.sin_port = htons(5000);// THis is used for host byte order to network byte order
  server.sin_addr.s_addr = INADDR_ANY;// THis says the server to bind to any of the client present on this local system
  bzero(&server.sin_zero,8);// this is filled with 8 zeros by bzero function


  myself.sin_family = AF_INET;// THis denotes the family of the server IPv4
  myself.sin_port = htons(50001);// THis is used for host byte order to network byte order
  myself.sin_addr.s_addr = INADDR_ANY;// THis says the server to bind to any of the client present on this local system
  bzero(&myself.sin_zero,8);// this is filled with 8 zeros by bzero function


  len = sizeof(struct sockaddr_in);
```

```
    char username [] = "personA\0";
    char pssd [] = "12345\0";
    char user[100];
    char psd[100];
    int k = 0;
    printf("Enter username:\n");
    scanf("%s",user);
    printf("Enter password:\n");
    scanf("%s",psd);
    printf("User name %s %d
%d\n",user,strlen(user),strlen(username));
    printf("Password %s\n",psd);
    if (strcmp(username,user)==0&&strcmp(pssd,psd)==0){
        printf("Successfully LOgged in!\n");

    // Binding will be done now

        if ((bind(sock,(struct sockaddr*)&myself,len))==-1){
            //Error checking in binding
            perror("Bind");
            exit(-1);
        }
        char send_buff [] = "a1";
```

```
sendto(sock,(char*)send_buff,strlen(send_buff),MSG_CONFIRM,(str
uct sockaddr*)&server,len);

    // Now we have entered recievefrom phase of the server



    while(1){

        // Niw we need to create a recieve and update from server
        pthread_t t1;
        pthread_t t2;
        pthread_create(&t1,NULL,recServer,NULL);

        sendToall(NULL);
        pthread_join(t1,NULL);


    }
}
else{
    printf("Check the entered username and password!\n");
}
```

```
   return 0;
}

// Client B

#include<stdio.h>
#include<stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include<netinet/in.h>// Used for network and node byte ordering
#include <errno.h>// this is useful if error is present then perror will be triggered from our code
#include<string.h>
#include<time.h>
#include<unistd.h>

#include<pthread.h>

#define MAX 1024



/*
First let us recall that we need to create a  socket
THen bind
```

THen we need to recievefrom in case of UDP

Then we need to sendfrom in case of UDP

*/

/*

sockaddr_in has the following predefined

sin_family -> denoting which domain we will be using

sin_port -> the port number

sin_addr (of type in_addr) -> IP address

sin_zero of 8 bits just to pad

whereas

sockaddr is the one which is accepted by the API Calls

sa_family -> denoting to which domain it will belong to

sa_data -> it is the socket(port+IP address)

in_addr is another data structure having

s_addr -> where IP address is entered

*/

```
int sock, cli;// these are server and client descriptors they will be created when we etsablish sockets

int len = sizeof(struct sockaddr_in);// to get length of sockaddr_in structure

struct sockaddr_in server, client,myself,temp,seen,p; // declaring sockaddr_in structure variable as server and client by which we will be connecting

int count = 0;

struct table{
   int i ;
   struct sockaddr_in client;
};


char buffer [MAX];


int n= 0;


char recBuff [MAX];


struct table arr[10];// creating by default 10 clients at max
```

```
void* sendToall(void* args){
   printf("Enter the message to send\n");
   while((buffer[n++]=getchar())!='\n');
   for(int i =0;i<count;i++){
      sendto(sock,(char*)buffer,strlen(buffer),MSG_CONFIRM,(struct
sockaddr*)&arr[i].client,len);
   }

}

void* recServer(void *args){
   recvfrom(sock,(char*)recBuff,MAX,MSG_WAITALL,(struct
sockaddr*)&client,&len);
   if (client.sin_port == server.sin_port && seen.sin_port!=
client.sin_port){
      printf("Message recieved from server %d\n",atoi(recBuff));
      arr[count].i = count;
      temp.sin_family = AF_INET;// THis denotes the family of the
server IPv4
      temp.sin_port = atoi(recBuff);// THis is used for host byte order
to network byte order
      temp.sin_addr.s_addr = INADDR_ANY;// THis says the server to
bind to any of the client present on this local system
```

```
        bzero(&temp.sin_zero,8);
        arr[count].client = temp;
        count++;
    }
    else if(seen.sin_port!=client.sin_port){
        printf("Message recieved from, %d :
%s\n",client.sin_port,recBuff);
    }
    else{
        return;
    }
    sleep(1);
}




int main(){
    // WE will be creating a client with bind so that we won't have any
confusion


    if((sock = socket(AF_INET,SOCK_DGRAM,0))==-1){
        // THis provides error checking and we will exit if socket is not
created
        perror("Socket");
```

```
    exit(-1);
  }


  server.sin_family = AF_INET;// THis denotes the family of the server IPv4
  server.sin_port = htons(5000);// THis is used for host byte order to network byte order
  server.sin_addr.s_addr = INADDR_ANY;// THis says the server to bind to any of the client present on this local system
  bzero(&server.sin_zero,8);// this is filled with 8 zeros by bzero function


  myself.sin_family = AF_INET;// THis denotes the family of the server IPv4
  myself.sin_port = htons(50002);// THis is used for host byte order to network byte order
  myself.sin_addr.s_addr = INADDR_ANY;// THis says the server to bind to any of the client present on this local system
  bzero(&myself.sin_zero,8);// this is filled with 8 zeros by bzero function


  len = sizeof(struct sockaddr_in);


  char username [] = "personB\0";
```

```
    char pssd [] = "12345\0";
    char user[100];
    char psd[100];
    int k = 0;
    printf("Enter username:\n");
    scanf("%s",user);
    printf("Enter password:\n");
    scanf("%s",psd);
    printf("User name %s %d
%d\n",user,strlen(user),strlen(username));
    printf("Password %s\n",psd);
    if (strcmp(username,user)==0&&strcmp(pssd,psd)==0){
        printf("Successfully LOgged in!\n");

    // Binding will be done now

        if ((bind(sock,(struct sockaddr*)&myself,len))==-1){
            //Error checking in binding
            perror("Bind");
            exit(-1);
        }
        char send_buff [] = "a2";
```

```
sendto(sock,(char*)send_buff,strlen(send_buff),MSG_CONFIRM,(str
uct sockaddr*)&server,len);

      // Now we have entered recievefrom phase of the server




      //bzero(&server.sin_zero,8);// this is filled with 8 zeros by bzero
function
      while(1){

         // Niw we need to create a recieve and update from server
         pthread_t t1;
         pthread_t t2;
         pthread_create(&t1,NULL,recServer,NULL);

         sendToall(NULL);
         pthread_join(t1,NULL);



      }
   }
   else{
      printf("Check the entered username and password!\n");
```

```
    }
    return 0;
}
```