Linux Module 1 – Linux Internals

Day 10 - Final Assessment

Group Chat Application Using C Program

Client:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>                 //for string operations
#include <unistd.h>                 //NULL constant defined here
#include <sys/types.h>
#include <sys/socket.h>             //for sockets
#include <netinet/in.h>             //Internet Protocol family sockaddr_in defined here
#include <pthread.h>                // for the cosy POSIX threads
#include <signal.h>                 //for ctrl+c signal
#include <stdio_ext.h>
#include <arpa/inet.h>
#define MYPORT 2012                 /* default port number */
#define MAXDATALEN 256


int     sockfd;
int     n,x;                        /*variables for socket*/
struct sockaddr_in serv_addr;       /* structure to hold server's address */
char    buffer[MAXDATALEN];
char    buf[10];
```

```
void *quitproc();
void* chat_write(int);
void* chat_read(int);
void *zzz();
int main(int argc, char *argv[])
{
pthread_t thr1,thr2;
if( argc != 2 )
        {
        printf("help:u need to put server ip\n");
                exit(0);
        }
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd == -1)
        printf ("client socket error\n");
else
        printf("socket\t\tcreated\n");
bzero((char *) &serv_addr, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
serv_addr.sin_port = htons(MYPORT);
serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
bzero(buf,10);
printf("\nENTER YOUR NAME::");
fgets(buf,10,stdin);
```

```
__fpurge(stdin);
buf[strlen(buf)-1]=':';
if(connect(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr))==-1)
        {
                printf("client connect error\n");
                exit(0);
        }
else
        printf("%s connected to server\n",buf);
printf("\rYOU JOINED AS- %s",buffer-1);
send(sockfd,buf,strlen(buf),0);
pthread_create(&thr2,NULL,(void *)chat_write,(void *)sockfd);
pthread_create(&thr1,NULL,(void *)chat_read,(void *)sockfd);
pthread_join(thr2,NULL);
pthread_join(thr1,NULL);
return 0;
}

void* chat_read(int sockfd)
{
if (signal(SIGINT,(void *)quitproc)==0)
if(signal(SIGTSTP, (void *)zzz)==0)
while(1)
{
```

```
n=recv(sockfd,buffer,MAXDATALEN-1,0);
        if(n==0){
             printf("\nDUE TO SOME UNEXPECTED REASONS SERVER HAS BEEN SHUTDOWN\n\n");
          exit(0);
          }

        if(n>0){
            printf("\n%s ",buffer);
         bzero(buffer,MAXDATALEN);
          }
   }//while ends

}


void* chat_write(int sockfd)

{
   while(1)
   {
       printf("%s",buf);
          fgets(buffer,MAXDATALEN-1,stdin);
```

```
          if(strlen(buffer)-1>sizeof(buffer))
          {
           printf("buffer size full \t enter within %lu characters \n",sizeof(buffer));
           bzero(buffer,MAXDATALEN);
           __fpurge(stdin);
           }

        n=send(sockfd,buffer,strlen(buffer),0);

          if(strncmp(buffer,"quit",4)==0)
           exit(0);

         bzero(buffer,MAXDATALEN);
    }//while ends
}

void *quitproc(){          //handling ctrl+d
      printf("\rPLEASE TYPE 'quit' TO EXIT\n");

}

void *zzz(){            //handling ctrl+z
      printf("\rPLEASE TYPE 'quit' TO EXIT\n");
```

```
}
```

Server:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>        //for sockets
#include <sys/types.h>
#include <string.h>        //for string operations
#include <netinet/in.h>        //Internet Protocol family sockaddr_in defined here
#include <pthread.h>        // for the cosy POSIX threads
#include <arpa/inet.h>        // for inet_ntoa() function
#include <unistd.h>        //NULL constant defined here
#include <signal.h>        //for ctrl+c signal
#define BACKLOG 100        // connections in the queue
#define MAXDATALEN 256        //max size of messages to be sent
#define PORT 2012        //default port number
struct Node                    /*structure to handle all clients*/
{
        int port;
        char username[10];
        struct Node *next;
};
typedef struct Node *ptrtonode;
typedef ptrtonode head;
typedef ptrtonode addr;
```

```
void sendtoall(char *,int new_fd);    /*send chat msgs to all connected clients*/
void Quitall( );                      /*send msg to all if server shuts down*/
head MakeEmpty( head h );         /*clearing list*/
void Delete( int port, head h );      /*delete client values on client exit*/
void Insert(int port,char*,head h,addr a);/*inserting new client */
void DeleteList( head h );            /*clearing list*/
void Display( const head h );       /*list all clients connected*/
void *Quitproc( );                     /*signal handler*/
void *server(void * arg);       /*server instance for every connected client*/
void zzz();
char     username[10];       /*size of username*/
int sf2;
head h;               /*variable of type struct head*/
char    buffer[MAXDATALEN];
/******main starts ***********/
int main(int argc, char *argv[])
{
int       sockfd,new_fd;            /*variables for socket*/
int     portnum;           /*variable for port numb if provided*/
struct sockaddr_in     server_addr;    /*structure to hold server's address */
struct sockaddr_in     client_addr;    /*structure to hold client's address */
int     cli_size,z;           /*length of address */
pthread_t     thr;           /*variable to hold thread ID */
int     yes=1;
```

```
addr a;                  /*variable of type struct addr*/
printf("\n\t*-*-*-*SERVER STARTED*-*-*-*\n");
/*=optional or default port argument=*/
if( argc == 2 )
portnum = atoi(argv[1]);
else
portnum = PORT;  //if port number not given as argument then using default port
printf("PORT NO.:\t%d\n",portnum);
h = MakeEmpty( NULL );        //frees the list
/*=set info of server =*/
server_addr.sin_family=AF_INET;        /* set family to Internet     */
server_addr.sin_addr.s_addr = htonl(INADDR_ANY); /* set IP address */
server_addr.sin_port=htons(portnum);
printf("IP ADDRESS:\t%s\n",inet_ntoa(server_addr.sin_addr));
/*=creating socket=*/
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if(sockfd == -1){
printf("server- socket() error");    // debugging
exit(1);
}else
printf("socket\t\tcreated.\n");
if (setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &yes,sizeof(int)) == -1) {
 printf("setsockopt error");    // debugging
exit(1);
```

```
}
else
printf("reusing\t\tport\n");
/*=binding socket=*/
if(bind(sockfd, (struct sockaddr *)&server_addr, sizeof(struct sockaddr))==-1){
printf("binding failed\n");
exit(1);}
else
printf("binding\t\tsuccess.\n\n");
printf("\t\tPRESS CTRL+z TO VIEW ONLINE CLIENTS\n\n");
/*=socket on listening mode=*/
listen(sockfd, BACKLOG);
printf("waiting for clients.....\n");
if (signal(SIGINT,(void *)Quitproc)==0)    //signal handler
if(signal(SIGTSTP, zzz)==0)            //signal handler
while(1){
cli_size=sizeof(struct sockaddr_in);
//cli_size necessary as an argument for pthread_create
new_fd = accept(sockfd, (struct sockaddr *)&client_addr,&cli_size);
//accepting connection from client
a =h ;
/*=sign in with name=*/
bzero(username,10);
if(recv(new_fd,username,sizeof(username),0)>0);
```

```
username[strlen(username)-1]=':';
printf("\t%d->%s JOINED chatroom\n",new_fd,username);
sprintf(buffer,"%s IS ONLINE\n",username);
Insert( new_fd,username, h, a );            //inserting newly accepted client socked fd in list
a = a->next;
/*=notify all clients about newly joining clients=*/
a = h ;
do{
a = a->next;
sf2 = a->port;
if(sf2!=new_fd)
send(sf2,buffer ,sizeof(buffer),0);
} while( a->next != NULL );
printf("server got connection from %s & %d\n\n",inet_ntoa(client_addr.sin_addr),new_fd);
// debugging
struct Node args;                    //struct to pass multiple arguments to server function
args.port=new_fd;
strcpy(args.username,username);
pthread_create(&thr,NULL,server,(void*)&args);          //creating thread for every client connected
pthread_detach(thr);
} /*while end*/
DeleteList(h);                    //deleting all clients when server closes
close(sockfd);
```

```
}
/* ==========Server function for every connected Client =========*/
void *server(void * arguments){
struct Node *args=arguments;
char    buffer[MAXDATALEN],ubuf[50],uname[10];    /* buffer for string the server sends
*/
char *strp;
char    *msg = (char *) malloc(MAXDATALEN);
int    ts_fd,x,y;
int    sfd,msglen;
ts_fd = args->port;    /*socket variable passed as arg*/
strcpy(uname,args->username);
addr    a;
/*=sending list of clients online=*/
a =h ;
do{
a = a->next;
sprintf( ubuf," %s is online\n",a->username );
send(ts_fd,ubuf,strlen(ubuf),0);
} while( a->next != NULL );
/*=start chatting=*/
while(1){
bzero(buffer,256);
y=recv(ts_fd,buffer,MAXDATALEN,0);
```

```
if (y==0)
goto jmp;
/*=if a client quits=*/
if ( strncmp( buffer, "quit", 4) == 0 ){
jmp:    printf("%d ->%s left chat deleting from list\n",ts_fd,uname);
sprintf(buffer,"%s has left the chat\n",uname);
addr a =  h ;
do{
a = a->next;
sfd = a->port;
            if(sfd == ts_fd)
              Delete( sfd, h );
            if(sfd != ts_fd)
              send(sfd,buffer,MAXDATALEN,0);
          }while ( a->next != NULL );

            Display( h );

            close(ts_fd);
            free(msg);

           break;
         }
```

```
/*=sending message to all clients =*/
printf("%s %s\n",uname,buffer);
strcpy(msg,uname);
x=strlen(msg);
strp = msg;
strp+= x;
strcat(strp,buffer);
msglen=strlen(msg);

   addr a = h ;
do{
        a = a->next;
        sfd = a->port;
if(sfd != ts_fd)
    send(sfd,msg,msglen,0);

} while( a->next != NULL );

  Display( h );
  bzero(msg,MAXDATALEN);

}//end while
 return 0;
```

```
}// end server



/*=====empties and deletes the list======*/
head MakeEmpty( head h )
    {
    if( h != NULL )
        DeleteList( h );
        h = malloc( sizeof( struct Node ) );
    if( h == NULL )
        printf( "Out of memory!" );
        h->next = NULL;
    return h;
    }
/*======delete list=======*/
void DeleteList( head h )
    {
    addr a, Tmp;
    a = h->next;
    h->next = NULL;
        while( a != NULL )
        {
        Tmp = a->next;
        free( a );
```

```
        a = Tmp;
        }
    }
/*===============inserting new clients to list==========*/
void Insert( int port,char *username, head h, addr a )
    {
    addr TmpCell;
    TmpCell = malloc( sizeof( struct Node ) );
        if( TmpCell == NULL )
        printf( "Out of space!!!" );
    TmpCell->port = port;
    strcpy(TmpCell->username,username);
    TmpCell->next = a->next;
    a->next = TmpCell;
    }

/*========displaying all clients in list==================*/
void Display( const head h )
    {
        addr a =h ;
        if( h->next == NULL )
    printf( "NO ONLINE  CLIENTS\n" );

        else
```

```
        {
        do
        {
            a = a->next;
            printf( "%d->%s \t", a->port,a->username );
        } while( a->next != NULL );
        printf( "\n" );
        }
    }

/*===========client deleted from list if client quits===============*/
void Delete( int port, head h ){
        addr a, TmpCell;
        a = h;
      while( a->next != NULL && a->next->port != port )
      a = a->next;
    if( a->next != NULL ){
    TmpCell = a->next;
    a->next = TmpCell->next;
    free( TmpCell );
    }
 }

/*======handling signals==========*/
```

```
void *Quitproc(){
    printf("\n\nSERVER SHUTDOWN\n");
    Quitall( );
        exit(0);
  }

/*===============notifying server shutdown==========*/
void Quitall(){
   int sfd;
   addr a = h ;
   int i=0;
     if( h->next == NULL ) {
       printf( "......BYE.....\nno clients \n\n" );
    exit(0);
       } else {

             do{
          i++;
             a = a->next;
             sfd = a->port;
             send(sfd,"server down",13,0);
             } while( a->next != NULL );
          printf("%d clients closed\n\n",i);
      }
```

```
}

void zzz(){
   printf("\rDISPLAYING ONLINE CLIENTS\n\n");
   Display(h);
}
```

Output: