ECHO SERVER

1. SERVER

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<unistd.h>

#define BUFLEN 1024 /* buffer length */
int main(int argc, char **argv)
{
int n;
int yes=1;
int sd, new_sd, client_len, port;
struct sockaddr_in server, client;
char buf[BUFLEN];
port = atoi(argv[1]);
/* create a stream socket */

if((sd = socket(AF_INET, SOCK_STREAM, 0)) == -1)
{
fprintf(stderr,"can't create a socket\n");
exit(1);
}
/* Fill the structure fileds with values */
server.sin_family = AF_INET;
```

```
server.sin_port = port;
server.sin_addr.s_addr =inet_addr("127.0.0.1");
// Reuse the port and address
if (setsockopt(sd, SOL_SOCKET, SO_REUSEADDR, &yes, sizeof(yes)) == -1) {
perror("setsockopt");
exit(1);
}
/* bind an address to the socket */
if(bind(sd, (struct sockaddr *)&server, sizeof(server)) == -1)
{
fprintf(stderr, "can't bind name to socket\n");
exit(1);
}
/* queue up to 5 connect requests */
listen(sd,5);
while(1)
{
client_len = sizeof(client);
if((new_sd = accept(sd, (struct sockaddr *) &client, &client_len)) == -1)
{
fprintf(stderr, "can't accept client \n");
exit(1);
}
n = read(new_sd, buf, sizeof(buf));

printf("The message received by client : %s\n",buf);

write(new_sd, buf,n);
close(new_sd);
```

```
}
close(sd);
return(0);
}
```

2. CLIENT

```
#include <stdio.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<unistd.h>

#define BUFLEN 1024 /* buffer length */
int main(int argc, char **argv)
{
int n;
int sd, port;
char buf[BUFLEN];
struct sockaddr_in server;
port=atoi(argv[1]);
/* create a stream socket */
if(( sd = socket(AF_INET, SOCK_STREAM, 0)) == -1)
{
fprintf(stderr, "can't create a socket\n");
exit(1);
```

```
}
// bzero((char *)&server, sizeof(struct sockaddr_in));
server.sin_family = AF_INET;
server.sin_port = port;
server.sin_addr.s_addr = inet_addr("127.0.0.1");

/* connecting to the server */
if(connect(sd, (struct sockaddr *)&server, sizeof(server)) == -1)
{
fprintf(stderr, "can't connect\n");
exit(1);
}
printf("Enter the message to be echoed: ");
scanf("%s",buf); /* get user's text */
write(sd, buf, BUFLEN); /* send it out */
printf("Echoed Messege:\n**************\n");
n = read(sd, buf, sizeof(buf));

printf("%s\n",buf);
close(sd);
return(0);
}
```

CHAT SERVR

1.SERVER

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
```

```
#include <netinet/in.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<unistd.h>
#include<string.h>

#define SERVER_TCP_PORT 6001 /* well known port */
#define BUFLEN 256 /* buffer length */
#define MAX 80
int flag=0;

int func(int sockfd)
{
char buff[MAX];
int n;

for(;;)
{
if(flag==1)
break;
bzero(buff,MAX);
n=read(sockfd,buff,sizeof(buff));

printf("Message from client is:%s",buff);
bzero(buff,MAX);
n=0;
//while((buff[n++]=getchar())!='\n');
printf("Enter message to be sent to client:\n");
fgets(buff,sizeof(buff),stdin);
```

```
n=strlen(buff);
if(strncmp("exit",buff,4)==0)
{
printf("Server Exit ...\n");
flag=1;
break;
}
else
{
write(sockfd,buff,sizeof(buff));
bzero(buff,MAX);
}

} // for loop

}

int main(int argc, char **argv)
{
int n;
int yes=1;
int sd, new_sd, client_len, port;
struct sockaddr_in server, client;
char buff[BUFLEN];

port = atoi(argv[1]);
// port=5750;

/* create a stream socket */
```

```
if((sd = socket(AF_INET, SOCK_STREAM, 0)) == -1)
{
fprintf(stderr,"can't create a socket\n");
exit(1);
}

/* bind an address to the socket */
// bzero((char *)&server, sizeof(struct sockaddr_in));
server.sin_family = AF_INET;
server.sin_port = port;
server.sin_addr.s_addr =htonl(INADDR_ANY);

if (setsockopt(sd, SOL_SOCKET, SO_REUSEADDR, &yes, sizeof(yes)) == -1) {
perror("setsockopt");
exit(1);
}

if(bind(sd, (struct sockaddr *)&server, sizeof(server)) == -1)
{
fprintf(stderr, "can't bind name to socket\n");
exit(1);
}

/* queue up to 5 connect requests */
listen(sd,5);

while(1)
{
client_len = sizeof(client);
```

```
if((new_sd = accept(sd, (struct sockaddr *) &client, &client_len)) == -1)
{
fprintf(stderr, "can't accept client\n");
exit(1);
}

func(new_sd);
close(new_sd);

}

close(sd);
return(0);
}
```

2.CLIENT

```
#include <stdio.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<unistd.h>
#include<string.h>

#define BUFLEN 256 /* buffer length */
#define MAX 80
```

```
void func(int sockfd)
{
char buff[MAX];
int n;
for(;;)
{
bzero(buff,sizeof(buff));
printf("Enter the message to be sent: ");
n=0;
fgets(buff,sizeof(buff),stdin);
if((strncmp(buff,"exit",4))==0)
{
printf("Client Exit...\n");
break;
}
n=strlen(buff);
write(sockfd,buff,n);
bzero(buff,sizeof(buff));
read(sockfd,buff,sizeof(buff));
printf("Message from Server : %s",buff);

}
}

int main(int argc, char **argv)
{
int n;
int sd, port;
char buff[BUFLEN];
```

```
struct sockaddr_in server;
//command line argument
port=atoi(argv[1]);
/* create a stream socket */
if(( sd = socket(AF_INET, SOCK_STREAM, 0)) == -1)
{
fprintf(stderr, "can't create a socket\n");
exit(1);
}

// bzero((char *)&server, sizeof(struct sockaddr_in));
server.sin_family = AF_INET;
server.sin_port = port;
server.sin_addr.s_addr = inet_addr("10.10.4.5");

/* connecting to the server */
if(connect(sd, (struct sockaddr *)&server, sizeof(server)) == -1)
{
fprintf(stderr, "can't connect\n");
exit(1);
}

func(sd);
close(sd);
return(0);
}
```

FTP

1.SERVER

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<unistd.h>
#include<string.h>

#define CHUNK 1024 /* read 1024 bytes at a time */

void readfile(int new_sd)
{
char buf[CHUNK];
int n;
FILE *file;

file = fopen("temp.txt", "r");
if(file ==NULL)
{
printf("\n Error in opening a file");
}
else
```

```
{
while(fgets(buf, sizeof(buf), file) !=NULL)

{
n=strlen(buf);
write(new_sd, buf,n);
bzero(buf, sizeof(buf));
}
}
fclose(file);
}

int main(int argc, char **argv)
{
int n;
int yes=1;
int sd, new_sd, client_len, port;
struct sockaddr_in server, client;
char buf[1024];

port = atoi(argv[1]);
// port=5750;

/* create a stream socket */
if((sd = socket(AF_INET, SOCK_STREAM, 0)) == -1)
{
fprintf(stderr,"can't create a socket\n");
exit(1);
}
```

```
/* bind an address to the socket */
// bzero((char *)&server, sizeof(struct sockaddr_in));
server.sin_family = AF_INET;

server.sin_port = port;
server.sin_addr.s_addr =inet_addr("127.0.0.1");

if (setsockopt(sd, SOL_SOCKET, SO_REUSEADDR, &yes, sizeof(yes)) == -1) {
perror("setsockopt");
exit(1);
}

if(bind(sd, (struct sockaddr *)&server, sizeof(server)) == -1)
{
fprintf(stderr, "can't bind name to socket\n");
exit(1);
}

/* queue up to 5 connect requests */
listen(sd,5);

while(1)
{
client_len = sizeof(client);

if((new_sd = accept(sd, (struct sockaddr *) &client, &client_len)) == -1)
{
fprintf(stderr, "can't accept client\n");
```

```
exit(1);
}

n = read(new_sd, buf, sizeof(buf));

//printf("The message received by client : %s \n",buf);

//write(new_sd, buf,n);
readfile(new_sd);

close(new_sd);

}

close(sd);
return(0);
}
```

2.CLIENT

```
#include <stdio.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<unistd.h>
#include<string.h>
```

```
#define CHUNK 1024

void writefile(int sd)
{
char buf[CHUNK];

int n,i;
FILE *file;

file = fopen("Output.txt", "w");
if(file==NULL)
{
printf("\n Error in opening a file");
}
while( (read(sd, buf, sizeof(buf))) > 0)
{

fputs(buf, file);
bzero(buf, sizeof(buf));
}

}

int main(int argc, char **argv)
{
int n;
int sd, port;
char buf[1024];
```

```
struct sockaddr_in server;

port=atoi(argv[1]);
/* create a stream socket */
if(( sd = socket(AF_INET, SOCK_STREAM, 0)) == -1)
{
fprintf(stderr, "can't create a socket\n");

exit(1);
}

// bzero((char *)&server, sizeof(struct sockaddr_in));
server.sin_family = AF_INET;
server.sin_port = port;
server.sin_addr.s_addr = inet_addr("127.0.0.1");

/* connecting to the server */
if(connect(sd, (struct sockaddr *)&server, sizeof(server)) == -1)
{
fprintf(stderr, "can't connect\n");
exit(1);
}

printf("\n Enter the command");
scanf("%s",buf); /* get user's text */
write(sd, buf, sizeof(buf)); /* send it out */
//printf("\n\nEchoed Messege:\n**************\n");

// n = read(sd, buf, sizeof(buf));
```

```
// printf("%s\n\n\n",buf);


writefile(sd);
close(sd);
return(0);
}
```