Basic TCP Client-Server programming on UNIX

Harshad B. Prajapati
Associate Professor
Information Technology Department,
Dharmsinh Desai University, Nadiad

Echo client-server application

- 1. The Client reads a line of text from its standard input and writes the line to the server.
- 2. The server reads the line from its network input and echoes the line back to the client.
- 3. The client reads the echoed line and prints it on its standard output.

echoTCPServer.c

```
/*This server program is concurrent*/
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <netinet/in.h>
#define MAXLINESIZE     100
#define SERV_PORT       5555
int listensd,clientsd;
char buffer[MAXLINESIZE+1];
struct sockaddr_in servaddr;
int noBytesRead=0;

/*this function will server client that connects*/
void processClient(int);
```

echoTCPServer.c

```
int main(){
    /*Create socket*/
    if((listensd=socket(AF_INET,SOCK_STREAM,0))<0){
        fprintf(stderr,"Cannot create socket\n");
        exit(-1);
    }
    /*Initialize socket address structure*/

    bzero(&servaddr,sizeof(servaddr));
    servaddr.sin_family=AF_INET;
    servaddr.sin_port=htons(SERV_PORT);
    servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
```

echoTCPServer.c

```
    /*bind socket address to the socket*/

    if(bind(listensd,(struct sockaddr*)&servaddr,sizeof(servaddr))<0){
        fprintf(stderr,"Error in bind\n");
        exit(-1);
    }
    /*Make the socket listening socket*/

    if(listen(listensd,5)<0){
        fprintf(stderr,"Error in listen\n");
        exit(-1);
    }
```

echoTCPServer.c

```
    for(;;){
        /*wait for client connection*/
        clientsd=accept(listensd,(struct sockaddr*)NULL,NULL);
        if(fork()==0){
                /*close listening socket in child. So that reference count
remains one.  The child serves the client. It does not need listening socket
to do this.     */
                close(listensd);
                /*server client*/
                processClient(clientsd);
                /*close connected socket*/
                close(clientsd);
                exit(0);
        }
```

echoTCPServer.c

```
        /*close connected socket in parent
        so that reference count remains one.
        */
        close(clientsd);
    }
    return 0;
}
void processClient(int clientsd){
    /*read message from client and send back*/
    while((noBytesRead=read(clientsd,buffer,sizeof(buffer)))>0)
        write(clientsd,buffer,noBytesRead);
}
```

echoTCPClient.c

```
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <netinet/in.h>
#include <string.h>
#define MAXLINESIZE     100
#define SERV_PORT       5555

int main(int argc,char** argv){
    int connectsd;
    char sendBuffer[MAXLINESIZE+1];
    char recvBuffer[MAXLINESIZE+1];
    struct sockaddr_in servaddr;
    int noBytesRead=0;
```

echoTCPClient.c

```
    if(argc!=2){
        fprintf(stderr,"Usage: %s IP-Address\n",argv[0]);
        exit(-1);
    }
    /*Create socket*/
    if((connectsd=socket(AF_INET,SOCK_STREAM,0))<0){
        fprintf(stderr,"Cannot create socket\n");
        exit(-1);
    }
    /*Initialize socket address structure*/
    bzero(&servaddr,sizeof(servaddr));
    servaddr.sin_family=AF_INET;
    servaddr.sin_port=htons(SERV_PORT);
```

echoTCPClient.c

```
    /*assign server address in socket address structure*/
    if(inet_pton(PF_INET,argv[1],&servaddr.sin_addr)<=0){
        fprintf(stderr,"Error in inet_pton\n");
        exit(-1);
    }
    /*Get connected with the server*/
    if(connect(connectsd,(struct sockaddr*)&servaddr,sizeof(servaddr))<0){
        fprintf(stderr,"Error in connect\n");
        exit(-1);
    }
```

echoTCPClient.c

```
    /*Read message from user through keyboard*/
    for(;gets(sendBuffer)!=NULL;){
        /*Send the message to the server*/
        write(connectsd,sendBuffer,strlen(sendBuffer)+1);
        if(noBytesRead=read(connectsd,recvBuffer,sizeof(recvBuffer))<0)
            exit(0);
        /*Display what the server sent in reply*/
        fprintf(stdout,"%s\n",recvBuffer);
    }
    return 0;
}
```

Echo client-server application

- Compile echoTCPServer.c and echoTCPClient.c using gcc.
- First run the server.
- Run clients and test.
- Exercises:
  - Understand how socket states of server and client sockets change.
  - Implement iterative version of the server. And test behavior/error conditions while using multiple clients.
- Important Note: Don't blindly copy programs from the text book. The text book uses wrapper functions and structures. These functions and structures start with capital letter (E.g, Bind rather that bind) and they are placed in user-defined header file unp.h that you will find included as #include "unp.h" (not <unp.h>).

Basic TCP Client-Server programming on UNIX

- We discussed up to 5.7 of chapter 5. TCP Client-Server Example of the text book.