EXPERIMENT - XIV
CONCURRENT FILE SERVER

April 6, 2020

ADITHYA D RAJAGOPAL
ROLL NO : 9
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
COLLEGE OF ENGINEERING TRIVANDRUM

AIM

To develop a concurrent file server which will provide the file requested by client if it exists. If not server sends appropriate message to the client. Server should also send its process ID (PID) to clients for display along with file or the message.

THEORY

File Transfer Protocol

FTP (File Transfer Protocol) is a client-server protocol that may be used to transfer files between computers on the internet. The client asks for the files and the server provides them. It is a standard network protocol used for the transfer of computer files between a client and server on a computer network. FTP is built on a client-server model architecture and uses separate control and data connections between the client and the server.FTP users may authenticate themselves with a clear-text sign-in protocol, normally in the form of a username and password, but can connect anonymously if the server is configured to allow it.

Server

Server should accept the incoming connection requests and establish a connection with the client. After this, it should check if the corresponding requested file is present or not. If found, the file is to be returned. Else the appropriate message is to be returned. The process ID is also to be sent to the client.

Client

After proper connection is established, the client should send the name of the file to be searched for. If found, the client should be able to receive the file as well as the PID of the server.

ALGORITHM

A model algorithm for the client and server is as follows:

Algorithm 1 Algorithm for the client

START
Create the socket using the function socket().
Configure socket details.
Connect the socket to server using function connect().
Send the name of the file to search for to the server.
if found **then**
    Receive the file.
else
    Console out the appropriate message.
end if
Receive the server's PID.
Terminate the connection.
STOP

Algorithm 2 Algorithm for the server

START
Configure socket details.
Bind the address struct to the socket using bind().
Listen to the socket.
A new socket for the incoming connection is created using accept().
Receive the name of the file to search for.
Check for the file using access().
if the file is found **then**
    Send the file to the client using sendfile().
else
    Send the response "NO" to the client.
end if
Send the PID to the client.
STOP

SOURCE CODE

Client

```
#include<stdio.h>
#include<sys/socket.h>
#include<stdlib.h>
#include<string.h>
#include<arpa/inet.h>
#include<unistd.h>
#include<fcntl.h>
#define PORT 8080
#define SA struct sockaddr

void main()
{
 int sockfd,connfd,filesize,fd;
 struct sockaddr_in servaddr;
 char *data,file[80],reply[80],ack[10];
 strcpy(ack,"Received");
 sockfd=socket(AF_INET,SOCK_STREAM,0);
 if(sockfd<0)
 {
  printf("Socket creation failed...\n");
  exit(0);
 }
 else
 printf("Socket successfully created...\n");
 bzero(&servaddr,sizeof(servaddr));
 servaddr.sin_family=AF_INET;
 servaddr.sin_addr.s_addr=inet_addr("127.0.0.1");
 servaddr.sin_port=htons(PORT);
 connfd=connect(sockfd,(SA*)&servaddr,sizeof(servaddr));
 if(connfd<0)
 {
```

```
  printf("Connection to the server failed...\n");
  exit(0);
 }
 else
 printf("Connected to the server...\n");
 printf("Enter the name of the file to search for : ");
 scanf("%s",file);
 write(sockfd,file,sizeof(file));
 bzero(reply,sizeof(reply));
 read(sockfd,reply,sizeof(reply));
 if(strcmp(reply,"NO")==0)
 printf("File not found...\n");
 else
 {
  printf("File found...\n");
  printf("Enter the name of the file to store to : ");
  scanf("%s",file);
  read(sockfd,&filesize,sizeof(int));
  data=malloc(filesize);
  fd=open(file,O_CREAT|O_EXCL|O_WRONLY,0666);
  recv(sockfd,data,filesize,0);
  write(fd,data,filesize);
  printf("File successfully copied to %s...\n",file);
  write(sockfd,ack,sizeof(ack));
  close(fd);
 }
 bzero(reply,sizeof(reply));
 read(sockfd,reply,sizeof(reply));
 int pid=atoi(reply);
 printf("Process ID of server : %d\n",pid);
 close(sockfd);
}
```

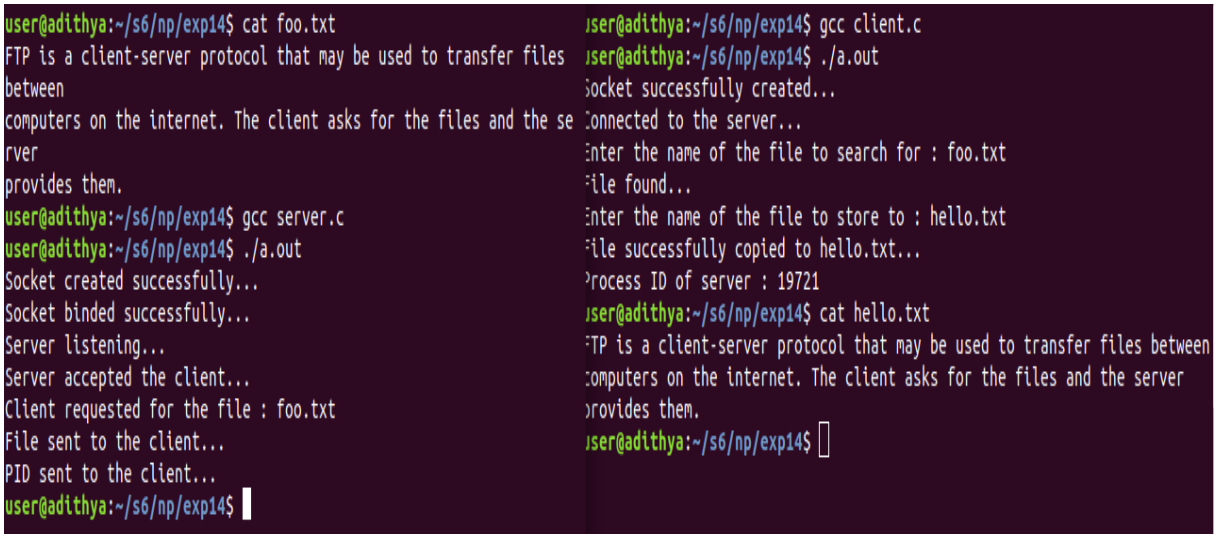
Server

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<arpa/inet.h>
#include<string.h>
#include<unistd.h>
#include<sys/sendfile.h>
#include<fcntl.h>
#define PORT 8080
#define SA struct sockaddr

int fsize(char *filename)
{
 FILE* fp=fopen(filename,"r");
 fseek(fp,0,2);
 int size=ftell(fp);
 return size;
}

void Xender(int sockfd)
{
 int pid;
 char file[80],response[80],ack[10];
 char fail[3],success[4];
 strcpy(fail,"NO");
 strcpy(success,"YES");
 read(sockfd,file,sizeof(file));
 printf("Client requested for the file : %s\n",file);
 if(access(file,F_OK)==-1)
 write(sockfd,fail,sizeof(fail));
 else
 {
  write(sockfd,success,sizeof(success));
```

```
   int fd,filesize;
   fd=open(file,O_RDONLY);
   filesize=fsize(file);
   write(sockfd,&filesize,sizeof(int));
   sendfile(sockfd,fd,NULL,filesize);
   printf("File sent to the client...\n");
   read(sockfd,ack,sizeof(ack));
  }
  pid=getpid();
  sprintf(response,"%d",pid);
  write(sockfd,response,sizeof(response));
  printf("PID sent to the client...\n");
}

void main()
{
  int sockfd,connfd,bindfd,len;
  struct sockaddr_in servaddr,cliaddr;
  sockfd=socket(AF_INET,SOCK_STREAM,0);
  if(sockfd<0)
  {
   printf("Socket creation failed...\n");
   exit(0);
  }
  else
  printf("Socket created successfully...\n");
  bzero(&servaddr,sizeof(servaddr));
  bzero(&cliaddr,sizeof(cliaddr));
  servaddr.sin_family=AF_INET;
  servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
  servaddr.sin_port=htons(PORT);
  bindfd=bind(sockfd,(SA*)&servaddr,sizeof(servaddr));
  if(bindfd<0)
  {
   printf("Socket bind failed...\n");
```

```
  exit(0);
 }
 else
 printf("Socket binded successfully...\n");
 listen(sockfd,3);
 printf("Server listening...\n");
 len=sizeof(cliaddr);
 connfd=accept(sockfd,(SA*)&cliaddr,&len);
 if(connfd<0)
 {
  printf("Server accept failed...\n");
  exit(0);
 }
 else
 printf("Server accepted the client...\n");
 Xender(connfd);
 close(sockfd);
}
```

OUTPUT


user@adithya:~/s6/np/exp14$ cat foo.txt
FTP is a client-server protocol that may be used to transfer files between
computers on the internet. The client asks for the files and the server
provides them.
user@adithya:~/s6/np/exp14$ gcc server.c
user@adithya:~/s6/np/exp14$ ./a.out
Socket created successfully...
Socket binded successfully...
Server listening...
Server accepted the client...
Client requested for the file : foo.txt
File sent to the client...
PID sent to the client...
user@adithya:~/s6/np/exp14$
user@adithya:~/s6/np/exp14$ gcc client.c
user@adithya:~/s6/np/exp14$ ./a.out
Socket successfully created...
Connected to the server...
Enter the name of the file to search for : foo.txt
File found...
Enter the name of the file to store to : hello.txt
File successfully copied to hello.txt...
Process ID of server : 19721
user@adithya:~/s6/np/exp14$ cat hello.txt
FTP is a client-server protocol that may be used to transfer files between
computers on the internet. The client asks for the files and the server
provides them.
user@adithya:~/s6/np/exp14$

RESULT

Developed a concurrent file server that provides the file requested by the client using C.