SSN COLLEGE OF ENGINEERING, KALAVAKKAM

(An Autonomous Institution, Affiliated to Anna University, Chennai)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UCS1511 – COMPUTER NETWORKS LAB

Lab Exercise 2b – FTP using TCP

Aim:

To transfer a file from server to client using TCP socket programming.

Algorithm:

Server:

1. Create a socket using socket() system call.
2. Bind the created socket with the port.
3. Listen for the connections.
4. When the server receives file name from the client, read the contents and send the contents to client.

Client:

1. Create a socket using socket() system call.
2. Connect it to the server.
3. Prompt the user to enter the file name.
4. Transfer the file name to the server
5. Receive the contents of the file and save in a new location
6. Close the socket

Code:

q2-server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h> // strcmp(), bzero()
#include <fcntl.h> // open()

#include <sys/types.h>  // socket(), bind(), listen(), accept()
#include <sys/socket.h> // socket(), inet_addr(), bind(), listen(), accept()

#include <netinet/in.h> // inet_addr()
#include <arpa/inet.h>  // htons(), inet_addr()

#include  <unistd.h> // read(), write(), close()

#include <errno.h> // to figure [errno]

#define PORT 9002
```

```
int main(int argc, char* argv[])
{
  // Check if args contain IPv4 address
  if(argc < 2)
  {
    printf("Usage: ./q2-server <IPv4-address>\n");
    return -1;
  }

  // create the server socket
  int server_socket;
  server_socket = socket(AF_INET, SOCK_STREAM, 0);

  if(server_socket == -1)
  {
    printf("Error in creating socket...\nerrno: %d\n", errno);
    return -1;
  }

  // define the server address
  struct sockaddr_in server_address;

  // clear the address pointer
  bzero(&server_address, sizeof(server_address));

  server_address.sin_family = AF_INET;
  server_address.sin_port = htons(PORT);
  // server_address.sin_addr.s_addr = INADDR_ANY;
  server_address.sin_addr.s_addr =  inet_addr(argv[1]);

  if(server_address.sin_addr.s_addr == -1)
  {
    printf("IPv4 Address Error...\nerrno: %d\n", errno);
    return -1;
  }

  // bind the socket to the specified IP and port
  if(bind(server_socket, (struct sockaddr*) &server_address,
sizeof(server_address)) == -1)
  {
    printf("Binding failed...\nerrno: %d\n", errno);
    return -1;
  }

  // listen for socket connection, max 5 in queue
  if(listen(server_socket, 5) == -1)
  {
    printf("Error while listening ...\nerrno: %d\n", errno);
    return -1;
  }

  int client_socket;
  client_socket = accept(server_socket, NULL, NULL);

  if(client_socket == -1)
  {
    printf("Error while accepting socket connection ...\nerrno: %d\n", errno);
    return -1;
```

```
  }

  printf("Waiting for client :/ ...\n");

  char buffer[256];

  read(client_socket, &buffer, sizeof(buffer));
  printf("File name received...\n");

  int fd = open(buffer, O_RDONLY, 0);
  if(fd < 1)
  {
    printf("File doesn't exist :( ...\n");
    write(client_socket, NULL, 0);
  }
  else
  {
    int read_bytes = 0;

    while(read_bytes = read(fd, &buffer, sizeof(buffer)))
    {
      write(client_socket, buffer, read_bytes);
      bzero(&buffer, sizeof(buffer));
    }

    printf("File transferred :) ...\n");
    close(fd);
  }

  // close the socket
  close(server_socket);

  return 0;
}
```

q2-client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h> // strcmp(), bzero()
#include <fcntl.h> // open()

#include <sys/types.h>  // socket(), connect()
#include <sys/socket.h> // socket(), inet_addr(), connect()

#include <netinet/in.h> // inet_addr()
#include <arpa/inet.h>  // htons(), inet_addr()

#include  <unistd.h> // read(), write(), close()

#include <errno.h> // to figure [errno]

#define PORT 9002

char* newFileName(char file[], char addnStr[]);

int main(int argc, char* argv[])
{
  // Check if args contain IPv4 address
```

```
  if(argc < 2)
  {
    printf("Usage: ./q2-client <IPv4-address>\n");
    return -1;
  }

  // create a socket
  int network_socket;

  network_socket = socket(AF_INET, SOCK_STREAM, 0);
  // AF_INET - Domain[Protocol Family](IPv4 Internet protocols)
  // SOCK_STREAM - TCP Socket
  // 0 - Protocol Specification (not a raw socket)

  if(network_socket == -1)
  {
    printf("Error in creating socket...\nerrno: %d\n", errno);
    return -1;
  }

  // specify the address for the socket
  struct sockaddr_in server_address;

  // clear the address pointer
  bzero(&server_address, sizeof(server_address));

  server_address.sin_family = AF_INET; // same family as our socket
  server_address.sin_port = htons(PORT); // set a port to listen to
  // server_address.sin_addr.s_addr =  INADDR_ANY; // INADDR_ANY = 0.0.0.0
  server_address.sin_addr.s_addr =  inet_addr(argv[1]); // convert a numbers and
dots TO byte order

  if(server_address.sin_addr.s_addr == -1)
  {
    printf("IPv4 Address Error...\nerrno: %d\n", errno);
    return -1;
  }

  // check for error in connection
  if (connect(network_socket, (struct sockaddr *) &server_address,
sizeof(server_address)) == -1) {
    printf("Error in connecting to server :/ Try again:(...\n");
    close(network_socket);
    return -1;
  }
  else {
    printf("Connected :) ...\n");
  }

  char buffer[256];
  char fileName[256];
  int bytes, fd;

  bzero(buffer, sizeof(buffer));

  printf("Enter the path of the file: ");
  scanf(" %[^\n]%*c", fileName);
  write(network_socket, fileName, sizeof(fileName));
```

```
  bytes = read(network_socket, &buffer, sizeof(buffer));

  if(bytes == 0)
  {
    printf("File doesn't exist :( ... can not find on server...\n");
  }
  else
  {
    fd = open(newFileName(fileName, "_new"), O_CREAT | O_WRONLY, S_IRWXU);
    do
    {
      // printf("\n\nMessage from server----------\n");
      write(fd, &buffer, bytes);
    }while(bytes = read(network_socket, &buffer, sizeof(buffer)));
    close(fd);

    printf("Downloaded to %s\n", newFileName(fileName, "_new"));
  }


  // close the socket
  close(network_socket);

  return 0;
}

char* newFileName(char file[], char addnStr[])
{
  char* fileName;
  fileName = malloc(sizeof(strlen(file) + strlen(addnStr)));

  int j = 0;
  for(int i = 0; i <= strlen(file); i++, j++)
  {
    if(file[i] == '.')
    {
      for(int k = 0; k < strlen(addnStr); k++, j++)
      {
        fileName[j] = addnStr[k];
      }
    }
    fileName[j] = file[i];
  }
  return fileName;
}
```

Output:

samplefile:

```
aravind@aravind-VirtualBox:~/UCS1511/Ex2$ cat samplefile.txt
The first oranges weren't orange
There's only one letter that doesn't appear in any U.S. state name
A cow-bison hybrid is called a "beefalo"
Johnny Appleseed's fruits weren't for eating
Scotland has 421 words for "snow"
The "Windy City" name has nothing to do with Chicago weather
Peanuts aren't technically nuts
Armadillo shells are bulletproof
Firefighters use wetting agents to make water wetter
The longest English word is 189,819 letters long
```

Server:

```
aravind@aravind-VirtualBox:~/UCS1511/Ex2$ ./q2-server
Usage: ./q2-server <IPv4-address>
```

```
aravind@aravind-VirtualBox:~/UCS1511/Ex2$ ./q2-server 127.0.1.2
Waiting for client :/ ...
File name received...
File doesn't exist :( ...
aravind@aravind-VirtualBox:~/UCS1511/Ex2$ ./q2-server 127.0.1.2
Waiting for client :/ ...
File name received...
File transferred :) ...
```

Client:

```
aravind@aravind-VirtualBox:~/UCS1511/Ex2$ ./q2-client
Usage: ./q2-client <IPv4-address>
aravind@aravind-VirtualBox:~/UCS1511/Ex2$ ./q2-client 127.0.1.2
Connected :) ...
Enter the path of the file: sample.txt
File doesn't exist :( ... can not find on server...
aravind@aravind-VirtualBox:~/UCS1511/Ex2$ ./q2-client 127.0.1.2
Connected :) ...
Enter the path of the file: samplefile.txt
Downloaded to samplefile_new.txt
```

samplefile_new.txt:

```
aravind@aravind-VirtualBox:~/UCS1511/Ex2$ cat samplefile_new.txt
The first oranges weren't orange
There's only one letter that doesn't appear in any U.S. state name
A cow-bison hybrid is called a "beefalo"
Johnny Appleseed's fruits weren't for eating
Scotland has 421 words for "snow"
The "Windy City" name has nothing to do with Chicago weather
Peanuts aren't technically nuts
Armadillo shells are bulletproof
Firefighters use wetting agents to make water wetter
The longest English word is 189,819 letters long
```

Learning outcomes:

1. Learned to transfer file contents using TCP connection.
2. Explored use of socket(), bind(), listen(), accept() on the server side and connect() on the client side.
3. Revisited file handling through open(), read() and write().
4. Understood importance of write() and read() in network communication.
5. Explored a few errno.