MANIPAL ACADEMY of HIGHER EDUCATION (Institution of Eminence Deemed to be University)

Manipal Institute of Technology (MIT), Bengaluru

Computer Networks (CSE_3152) Assignment Sem: V

CSE Section: A

Max Marks: 10

Date of Announcement: 19-10-2024 Date-of-Submission: 29-10-2024

Computer Networks (CSE_3152) – Assignment							
Registration Number	Name	Program	Section	Assignment Marks			
225805222	RUTVIK AVINASH BARBHAI	Bachelor of Technology (B.Tech)	CSE-CORE -A	Writeup		Viva (Based on Writeup)	
				Awarded	Max	Awarded	Max
					5		5

Assignment Questions :

I. Web Server (HTTP Communication)

Implementation:

Your(client) sends a request to a web server using HTTP. The server sends back the web page data, which the Client should display.

Ans:

SERVER

```
// server.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080
#define BUFFER_SIZE 4096
  int main() {
   int server_fd, new_socket;
   struct sockaddr_in address;
   int addrlen = sizeof(address);
   char buffer[BUFFER_SIZE] = {0};

   // HTML response (you can modify the HTML here)
   char *html_response = "HTTP/1.1 200 OK\r\n"
                  "Content-Type: text/html\r\n\r\n"
                  "<html><body><h1>Welcome to My Web Server!</h1></body></html>";
```

```
    // Create socket file descriptor
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
        perror("Socket failed");
        exit(EXIT_FAILURE);
    }

    // Set address structure
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);

    // Bind the socket to the network address
    if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {
        perror("Bind failed");
        close(server_fd);
        exit(EXIT_FAILURE);
    }

    // Listen for incoming connections
    if (listen(server_fd, 3) < 0) {
        perror("Listen failed");
        close(server_fd);
        exit(EXIT_FAILURE);
    }

    printf("Server is running on port %d...\n", PORT);

    // Accept incoming connection
    if ((new_socket = accept(server_fd, (struct sockaddr *)&address, (socklen_t *)&addrlen)) <
0) {
        perror("Accept failed");
        close(server_fd);
        exit(EXIT_FAILURE);
    }

    // Read the HTTP request from the client
    read(new_socket, buffer, BUFFER_SIZE);
    printf("Received request:\n%s\n", buffer);

    // Send the HTML response
    write(new_socket, html_response, strlen(html_response));

    // Close the socket
    close(new_socket);
    close(server_fd);
    return 0;
}
```

CLIENT

```
// client.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#define PORT 8080
#define BUFFER_SIZE 4096

int main() {
  int sock = 0;
  struct sockaddr_in serv_addr;
  char *request = "GET / HTTP/1.1\r\nHost: localhost\r\n\r\n";
  char buffer[BUFFER_SIZE] = {0};

  // Create socket
  if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    printf("\nSocket creation error\n");
    return -1;
  }

  // Set server address
  serv_addr.sin_family = AF_INET;
  serv_addr.sin_port = htons(PORT);

  // Convert IPv4 and IPv6 addresses from text to binary form
  if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) <= 0) {
    printf("\nInvalid address/Address not supported\n");
    return -1;
  }

  // Connect to server
  if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
    printf("\nConnection failed\n");
    return -1;
  }

  // Send HTTP GET request
  send(sock, request, strlen(request), 0);
  printf("HTTP request sent\n");

  // Read and print the response from the server
  read(sock, buffer, BUFFER_SIZE);
  printf("Received response:\n%s\n", buffer);

  // Close the socket
  close(sock);
  return 0;
}
```

II. File Transfer Application (FTP-like Service)

Implementation:

Client: Requests a file from the server or sends a file to the server. Client should display the downloaded file or display a message that file was uploaded on the server

Server: Listens for incoming file requests or uploads and processes them accordingly.

Ans:

Server (FTP-like File Transfer Server in C)

```
//  ftp_server.c
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <fcntl.h>

#define PORT 8080
#define BUFFER_SIZE 1024

int main() {
   int server_fd, new_socket, file;
   struct  sockaddr_in  address;
   char buffer[BUFFER_SIZE] = {0};

   server_fd = socket(AF_INET, SOCK_STREAM, 0);
   address.sin_family = AF_INET;
   address.sin_addr.s_addr = INADDR_ANY;
   address.sin_port = htons(PORT);

   bind(server_fd, (struct sockaddr *)&address, sizeof(address));
   listen(server_fd, 3);
   printf("Server listening...\n");

   new_socket = accept(server_fd, NULL, NULL);
   read(new_socket, buffer, BUFFER_SIZE); // Read client command

   if (strncmp(buffer, "GET", 3) == 0) {
      file = open("server_file.txt", O_RDONLY);
      read(file, buffer, BUFFER_SIZE);
      send(new_socket, buffer, strlen(buffer), 0); // Send file content
      close(file);
   } else if (strncmp(buffer, "PUT", 3) == 0) {
      recv(new_socket, buffer, BUFFER_SIZE, 0);
      file = open("server_upload.txt", O_WRONLY | O_CREAT, 0666);
      write(file, buffer, strlen(buffer)); // Write uploaded content
      close(file);
   }

   close(new_socket);
   close(server_fd);
   return 0;
}
```

Client (FTP-like File Transfer Client in C)

```
// ftp_client.c
#include <stdio.h>
```

```
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <fcntl.h>

#define PORT 8080
#define BUFFER_SIZE 1024

int main() {
  int sock = socket(AF_INET, SOCK_STREAM, 0), file;
  struct sockaddr_in serv_addr;
  char buffer[BUFFER_SIZE] = {0}, command[4];

  serv_addr.sin_family = AF_INET;
  serv_addr.sin_port = htons(PORT);
  inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr);

  connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr));

  printf("Enter command (GET/PUT): ");
  scanf("%s", command);
  send(sock, command, strlen(command), 0);  // Send command to server

  if (strcmp(command, "GET") == 0) {
    recv(sock, buffer, BUFFER_SIZE, 0);
    printf("File content:\n%s\n", buffer);  // Display downloaded file
  } else if (strcmp(command, "PUT") == 0) {
    file = open("client_upload.txt", O_RDONLY);
    read(file, buffer, BUFFER_SIZE);
    send(sock, buffer, strlen(buffer), 0); // Send file content to server
    printf("File uploaded to server.\n");
    close(file);
  }

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

III. Multiplayer Game (Peer-to-Peer Communication)

Implementation: Multiplayer (max players:5) online game use sockets to enable communication between players and a game server.

Server: Broadcasts secret string : H_ _ _O_ _ N to

other players in the game.

The secret string stored in the Server is : HALLOWEN which is not shared with other players

Other Players guess the secret word or string, character by character

Each participant gets the chance to guess as per the number of characters in the string which in the above example is 8 characters

Assume Player 1 guesses the next character following H as I; then Server should prompt it as wrong

Assume Player 2 guesses the next character following H as A; then Server should prompt that player as correct

The player or players who guess the word correctly are prompted as winner

Ans:
Server (Multiplayer Game Server in C)

```
// game_server.c
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080
#define MAX_PLAYERS 5
#define SECRET_WORD "HALLOWEEN"

void broadcast(char *message, int *clients, int num_players) {
  for (int i = 0; i < num_players; i++) {
    send(clients[i], message, strlen(message), 0);
  }
}

int main() {
  int server_fd, client_fds[MAX_PLAYERS], num_players = 0;
  struct sockaddr_in address;
  char buffer[1024] = {0}, guessed_word[] = "H_ _ _O_ _ _N";

  server_fd = socket(AF_INET, SOCK_STREAM, 0);
  address.sin_family = AF_INET;
  address.sin_addr.s_addr = INADDR_ANY;
  address.sin_port = htons(PORT);

  bind(server_fd, (struct sockaddr *)&address, sizeof(address));
  listen(server_fd, MAX_PLAYERS);
  printf("Waiting for players...\n");

  while (num_players < MAX_PLAYERS) {
    client_fds[num_players] = accept(server_fd, NULL, NULL);
    send(client_fds[num_players], guessed_word, strlen(guessed_word), 0);
    num_players++;
  }

  int guesses = 0;
  while (guesses < 8) {
    for (int i = 0; i < num_players; i++) {
      recv(client_fds[i], buffer, 1024, 0); // Get player's guess
      char guess = buffer[0];
      if (guess == SECRET_WORD[guesses]) {
        guessed_word[guesses * 2 + 1] = guess;
        broadcast(guessed_word, client_fds, num_players);  // Update all players
      } else {
        send(client_fds[i], "Wrong guess!", 12, 0);
      }

      if (strcmp(guessed_word, SECRET_WORD) == 0) {
        broadcast("We have a winner!", client_fds, num_players);
        goto end_game;
      }
      guesses++;
```

```
        }
    }

end_game:
    for (int i = 0; i < num_players; i++) {
        close(client_fds[i]);
    }
    close(server_fd);
    return 0;
}
```

Client (Multiplayer Game Client in C)

```
// game_client.c
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080

int main() {
    int sock = socket(AF_INET, SOCK_STREAM, 0);
    struct sockaddr_in serv_addr;
    char buffer[1024] = {0}, guess;

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);
    inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr);

    connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr));

    while (1) {
        recv(sock, buffer, 1024, 0);
        printf("%s\n", buffer);  // Display current word status or winner message

        if (strstr(buffer, "winner")) {
            break;  // End the game if there is a winner
        }

        printf("Enter your guess: ");
        scanf(" %c", &guess);
        send(sock, &guess, 1, 0);
    }

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