Name: Jayesh Bhikaji Pendharkar

Experiment No 5: Title- Socket Programming using C/C++/Java.
TCP Client, TCP Server
UDP Client, UDP Server

Program:

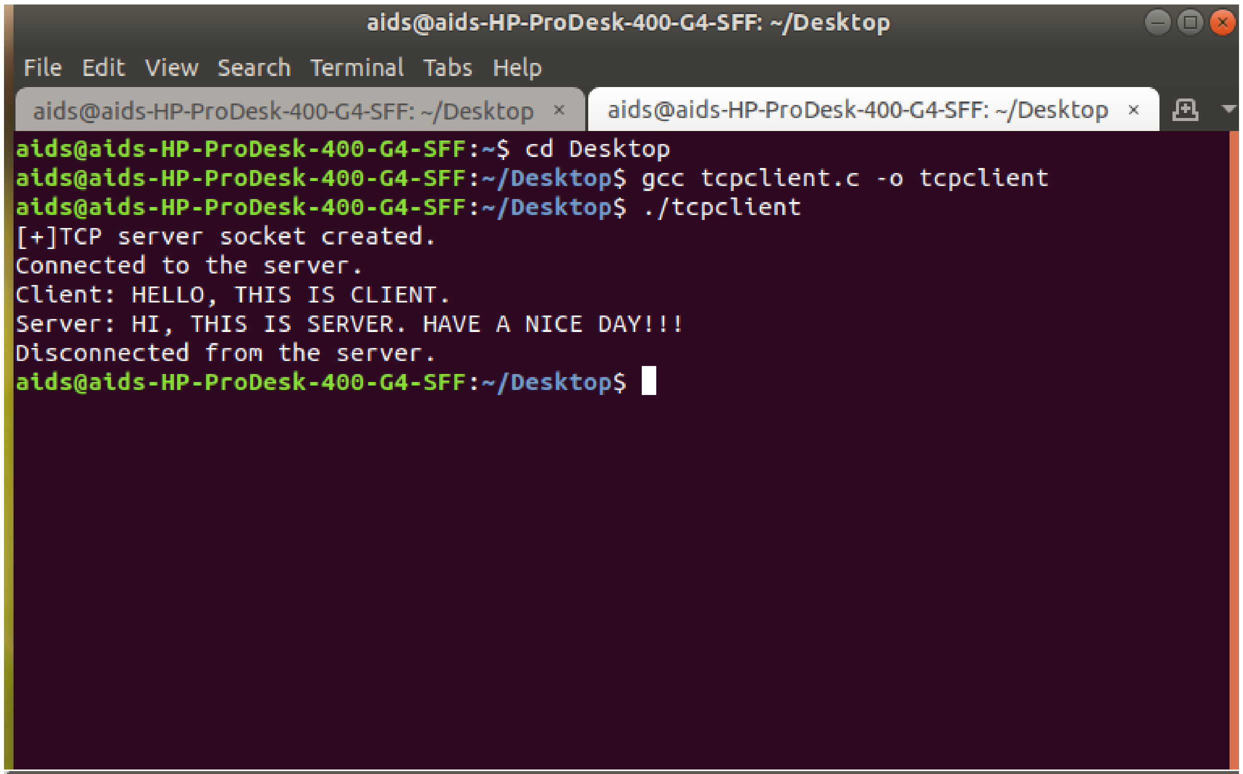
TCP Client

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

int main(){

  char *ip = "127.0.0.1";
  int port = 5566;

  int sock;
  struct sockaddr_in addr;
  socklen_t addr_size;
  char buffer[1024];
  int n;

  sock = socket(AF_INET, SOCK_STREAM, 0);
  if (sock < 0){
    perror("[-]Socket error");
    exit(1);
  }
  printf("[+]TCP server socket created.\n");

  memset(&addr, '\0', sizeof(addr));
  addr.sin_family = AF_INET;
  addr.sin_port = port;
  addr.sin_addr.s_addr = inet_addr(ip);

  connect(sock, (struct sockaddr*)&addr, sizeof(addr));
  printf("Connected to the server.\n");

  bzero(buffer, 1024);
  strcpy(buffer, "HELLO, THIS IS CLIENT.");
  printf("Client: %s\n", buffer);
  send(sock, buffer, strlen(buffer), 0);

  bzero(buffer, 1024);
  recv(sock, buffer, sizeof(buffer), 0);
  printf("Server: %s\n", buffer);

  close(sock);
  printf("Disconnected from the server.\n");

  return 0;

}
```

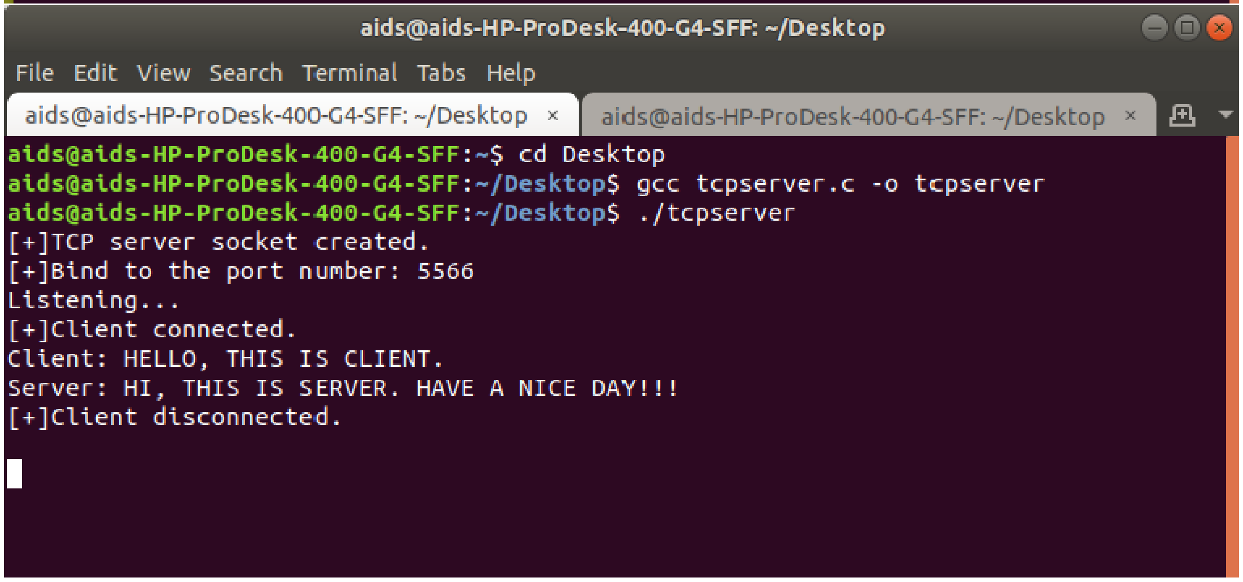
TCP Server

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

int main(){

  char *ip = "127.0.0.1";
  int port = 5566;

  int server_sock, client_sock;
  struct sockaddr_in server_addr, client_addr;
  socklen_t addr_size;
  char buffer[1024];
  int n;

  server_sock = socket(AF_INET, SOCK_STREAM, 0);
  if (server_sock < 0){
    perror("[-]Socket error");
    exit(1);
  }
  printf("[+]TCP server socket created.\n");

  memset(&server_addr, '\0', sizeof(server_addr));
  server_addr.sin_family = AF_INET;
  server_addr.sin_port = port;
  server_addr.sin_addr.s_addr = inet_addr(ip);

  n = bind(server_sock, (struct sockaddr*)&server_addr, sizeof(server_addr));
  if (n < 0){
    perror("[-]Bind error");
    exit(1);
  }
  printf("[+]Bind to the port number: %d\n", port);

  listen(server_sock, 5);
  printf("Listening...\n");

  while(1){
    addr_size = sizeof(client_addr);
    client_sock = accept(server_sock, (struct sockaddr*)&client_addr, &addr_size);
    printf("[+]Client connected.\n");

    bzero(buffer, 1024);
    recv(client_sock, buffer, sizeof(buffer), 0);
    printf("Client: %s\n", buffer);

    bzero(buffer, 1024);
    strcpy(buffer, "HI, THIS IS SERVER. HAVE A NICE DAY!!!");
    printf("Server: %s\n", buffer);
    send(client_sock, buffer, strlen(buffer), 0);

    close(client_sock);
    printf("[+]Client disconnected.\n\n");

  }

  return 0;
}
```

Output- TCP Client & Server :


aids@aids-HP-ProDesk-400-G4-SFF: ~/Desktop
File Edit View Search Terminal Tabs Help
aids@aids-HP-ProDesk-400-G4-SFF: ~/Desktop
aids@aids-HP-ProDesk-400-G4-SFF: ~/Desktop

```
aids@aids-HP-ProDesk-400-G4-SFF:~$ cd Desktop
aids@aids-HP-ProDesk-400-G4-SFF:~/Desktop$ gcc tcpclient.c -o tcpclient
aids@aids-HP-ProDesk-400-G4-SFF:~/Desktop$ ./tcpclient
[+]TCP server socket created.
Connected to the server.
Client: HELLO, THIS IS CLIENT.
Server: HI, THIS IS SERVER. HAVE A NICE DAY!!!
Disconnected from the server.
aids@aids-HP-ProDesk-400-G4-SFF:~/Desktop$
```


aids@aids-HP-ProDesk-400-G4-SFF: ~/Desktop
File Edit View Search Terminal Tabs Help
aids@aids-HP-ProDesk-400-G4-SFF: ~/Desktop
aids@aids-HP-ProDesk-400-G4-SFF: ~/Desktop

```
aids@aids-HP-ProDesk-400-G4-SFF:~$ cd Desktop
aids@aids-HP-ProDesk-400-G4-SFF:~/Desktop$ gcc tcpserver.c -o tcpserver
aids@aids-HP-ProDesk-400-G4-SFF:~/Desktop$ ./tcpserver
[+]TCP server socket created.
[+]Bind to the port number: 5566
Listening...
[+]Client connected.
Client: HELLO, THIS IS CLIENT.
Server: HI, THIS IS SERVER. HAVE A NICE DAY!!!
[+]Client disconnected.
```

Program:

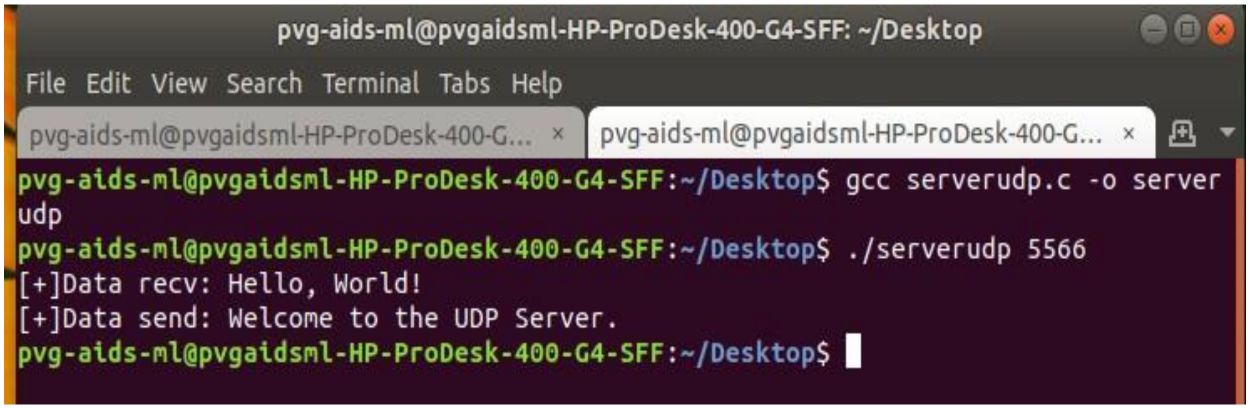
UDP Server

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

int main(int argc, char **argv){

  if (argc != 2){
    printf("Usage: %s <port>\n", argv[0]);
    exit(0);
  }

  char *ip = "127.0.0.1";
  int port = atoi(argv[1]);

  int sockfd;
  struct sockaddr_in server_addr, client_addr;
  char buffer[1024];
  socklen_t addr_size;
  int n;

  sockfd = socket(AF_INET, SOCK_DGRAM, 0);
  if (sockfd < 0){
    perror("[-]socket error");
    exit(1);
  }

  memset(&server_addr, '\0', sizeof(server_addr));
  server_addr.sin_family = AF_INET;
  server_addr.sin_port = htons(port);
  server_addr.sin_addr.s_addr = inet_addr(ip);

  n = bind(sockfd, (struct sockaddr*)&server_addr, sizeof(server_addr));
  if (n < 0) {
    perror("[-]bind error");
    exit(1);
  }

  bzero(buffer, 1024);
  addr_size = sizeof(client_addr);
  recvfrom(sockfd, buffer, 1024, 0, (struct sockaddr*)&client_addr, &addr_size);
  printf("[+]Data recv: %s\n", buffer);

  bzero(buffer, 1024);
  strcpy(buffer, "Welcome to the UDP Server.");
  sendto(sockfd, buffer, 1024, 0, (struct sockaddr*)&client_addr, sizeof(client_addr));
  printf("[+]Data send: %s\n", buffer);

  return 0;
}
```

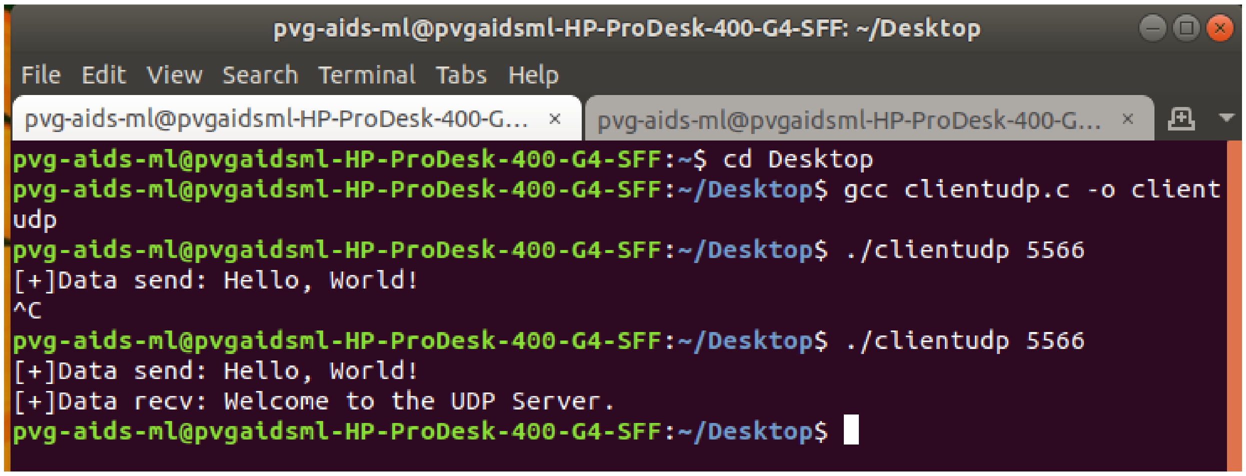
UDP Client:

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

int main(int argc, char **argv){

  if (argc != 2) {
    printf("Usage: %s <port>\n", argv[0]);
    exit(0);
  }

  char *ip = "127.0.0.1";
  int port = atoi(argv[1]);

  int sockfd;
  struct sockaddr_in addr;
  char buffer[1024];
  socklen_t addr_size;

  sockfd = socket(AF_INET, SOCK_DGRAM, 0);
  memset(&addr, '\0', sizeof(addr));
  addr.sin_family = AF_INET;
  addr.sin_port = htons(port);
  addr.sin_addr.s_addr = inet_addr(ip);

  bzero(buffer, 1024);
  strcpy(buffer, "Hello, World!");
  sendto(sockfd, buffer, 1024, 0, (struct sockaddr*)&addr, sizeof(addr));
  printf("[+]Data send: %s\n", buffer);

  bzero(buffer, 1024);
  addr_size = sizeof(addr);
  recvfrom(sockfd, buffer, 1024, 0, (struct sockaddr*)&addr, &addr_size);
  printf("[+]Data recv: %s\n", buffer);

  return 0;
}
```

Output- UDP Server & Client :


pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF: ~/Desktop
File Edit View Search Terminal Tabs Help
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G...
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G...

```
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF:~/Desktop$ gcc serverudp.c -o server
udp
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF:~/Desktop$ ./serverudp 5566
[+]Data recv: Hello, World!
[+]Data send: Welcome to the UDP Server.
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF:~/Desktop$
```


pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF: ~/Desktop
File Edit View Search Terminal Tabs Help
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G...
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G...

```
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF:~$ cd Desktop
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF:~/Desktop$ gcc clientudp.c -o client
udp
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF:~/Desktop$ ./clientudp 5566
[+]Data send: Hello, World!
^C
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF:~/Desktop$ ./clientudp 5566
[+]Data send: Hello, World!
[+]Data recv: Welcome to the UDP Server.
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF:~/Desktop$
```

Name: Jayesh Bhikaji Pendharkar

Experiment No 6: Title- Write a program using TCP socket for wired network for following
a.Say Hello to Each other
b.File transfer

TCP Socket Client:

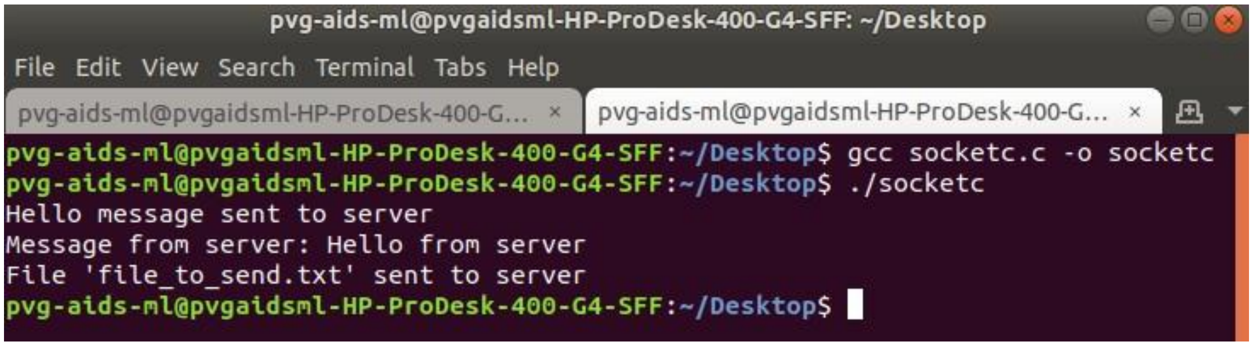
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080
#define BUFFER_SIZE 1024

void send_file(FILE *fp, int sockfd) {
    char data[BUFFER_SIZE] = {0};

    while (fgets(data, BUFFER_SIZE, fp) != NULL) {
        if (send(sockfd, data, sizeof(data), 0) == -1) {
            perror("Error sending file");
            exit(1);
        }
        bzero(data, BUFFER_SIZE);
    }
}

int main()  {
    int sock = 0;
    struct sockaddr_in serv_addr;
    char buffer[BUFFER_SIZE] = {0};
    char *hello = "Hello from client";
    FILE *fp;
    char *filename = "file_to_send.txt";

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

    // Define server address
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);
    // Convert IPv4 address to binary form
    if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) <= 0) {
        printf("\nInvalid address or 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 "Hello" message to the server
    send(sock, hello, strlen(hello), 0);
    printf("Hello message sent to server\n");

    // Receive "Hello" from server
    read(sock, buffer, BUFFER_SIZE);
    printf("Message from server: %s\n", buffer);

    // Send file to server
    fp = fopen(filename, "r");
    if (fp == NULL) {
        perror("File open error");
        exit(1);
    }

    send_file(fp, sock);
    printf("File '%s' sent to server\n", filename);

    fclose(fp);
    close(sock);

    return 0;
}
```

TCP Socket Client:

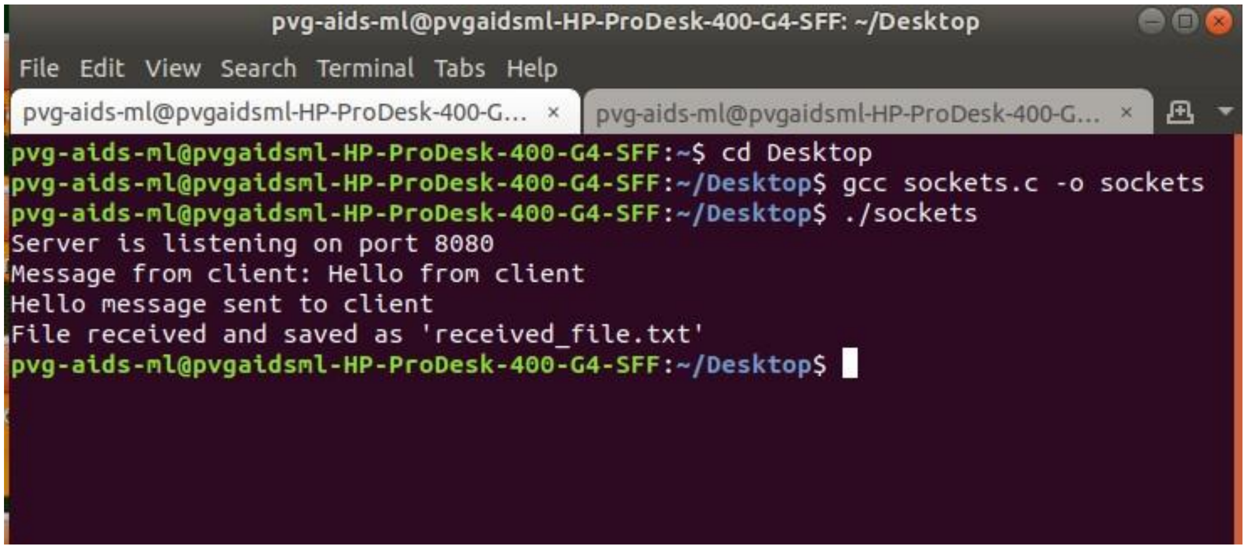
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080
#define BUFFER_SIZE 1024

void send_file(FILE *fp, int sockfd) {
    char data[BUFFER_SIZE] = {0};

    while (fgets(data, BUFFER_SIZE, fp) != NULL) {
        if (send(sockfd, data, sizeof(data), 0) == -1) {
            perror("Error sending file");
            exit(1);
        }
        bzero(data, BUFFER_SIZE);
    }
}

int main() {
    int server_fd, new_socket;
    struct sockaddr_in address;
    int addrlen = sizeof(address);
    char buffer[BUFFER_SIZE] = {0};
    char *hello = "Hello from server";
    FILE *fp;
    char *filename = "received_file.txt";

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

    // Define the server address
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);

    // Bind the socket to the 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 listening on port %d\n", PORT);

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

// Receive message from client
read(new_socket, buffer, BUFFER_SIZE);
printf("Message from client: %s\n", buffer);

// Send "Hello" back to the client
send(new_socket, hello, strlen(hello), 0);
printf("Hello message sent to client\n");

// Receive file from client
fp = fopen(filename, "w");
if (fp == NULL) {
    perror("File open error");
    exit(1);
}

while (1) {
    ssize_t n = recv(new_socket, buffer, BUFFER_SIZE, 0);
    if (n <= 0) {
        break;
    }
    fprintf(fp, "%s", buffer);
    bzero(buffer, BUFFER_SIZE);
}
printf("File received and saved as '%s'\n", filename);

fclose(fp);
close(new_socket);
close(server_fd);
```

```
    return 0;
}
```

Output- Client & Server:


pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF: ~/Desktop
File Edit View Search Terminal Tabs Help
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G...
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G...

```
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF:~/Desktop$ gcc socketc.c -o socketc
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF:~/Desktop$ ./socketc
Hello message sent to server
Message from server: Hello from server
File 'file_to_send.txt' sent to server
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF:~/Desktop$
```


pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF: ~/Desktop
File Edit View Search Terminal Tabs Help
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G...
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G...

```
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF:~$ cd Desktop
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF:~/Desktop$ gcc sockets.c -o sockets
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF:~/Desktop$ ./sockets
Server is listening on port 8080
Message from client: Hello from client
Hello message sent to client
File received and saved as 'received_file.txt'
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF:~/Desktop$
```

Name: Jayesh Bhikaji Pendharkar

Experiment No 7: Title - Write a program using UDP Sockets to enable file transfer (Script, Text, Audio and Video one file each) between two machines.

Program- UDP Client:

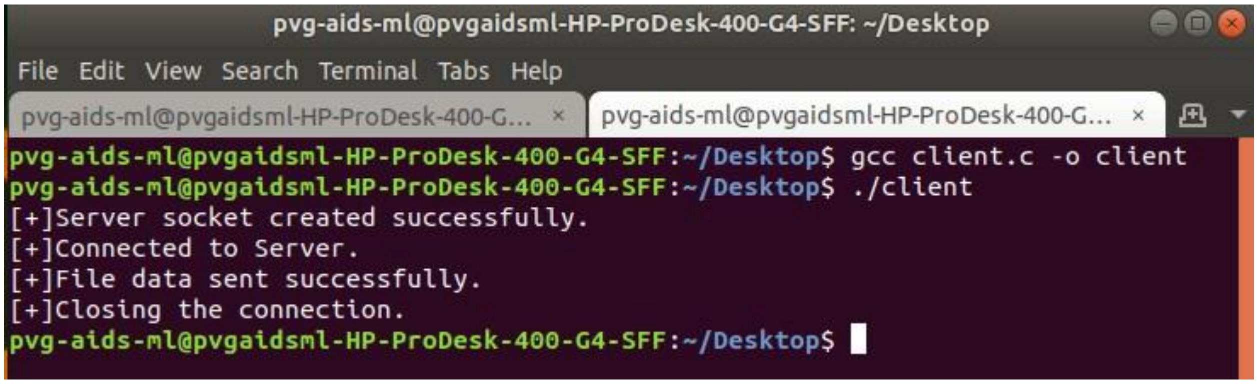
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080
#define BUFFER_SIZE 1024

void send_file(FILE *fp, int sockfd, struct sockaddr_in server_addr) {
    char data[BUFFER_SIZE] = {0};
    char ack[BUFFER_SIZE] = {0};
    socklen_t addr_len = sizeof(server_addr);
    ssize_t n;

    while (fgets(data, BUFFER_SIZE, fp) != NULL) {
        // Send file data to the server
        if (sendto(sockfd, data, strlen(data), 0, (struct sockaddr*)&server_addr, addr_len) == -1) {
            perror("Error sending file data");
            exit(1);
        }

        // Receive acknowledgment from the server
        n = recvfrom(sockfd, ack, BUFFER_SIZE, 0, (struct sockaddr*)&server_addr, &addr_len);
        if (n <= 0) {
            perror("Error receiving acknowledgment");
            exit(1);
        }

        bzero(data, BUFFER_SIZE);
        bzero(ack, BUFFER_SIZE);
    }

    // Send end-of-file signal to the server
    sendto(sockfd, "EOF", 3, 0, (struct sockaddr*)&server_addr, addr_len);
    printf("File sent successfully.\n");
}
```

```
int main() {
    int sockfd;
    struct sockaddr_in server_addr;
    FILE *fp;
    char *filename = "file_to_send.txt";  // Can replace with script, audio, or video file

    // Create UDP socket
    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    // Initialize server address
    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = INADDR_ANY;

    // Open file to send
    fp = fopen(filename, "r");
    if (fp == NULL) {
        perror("File open error");
        exit(1);
    }

    // Send file to server
    send_file(fp, sockfd, server_addr);

    // Close file and socket
    fclose(fp);
    close(sockfd);

    return 0;
}
```

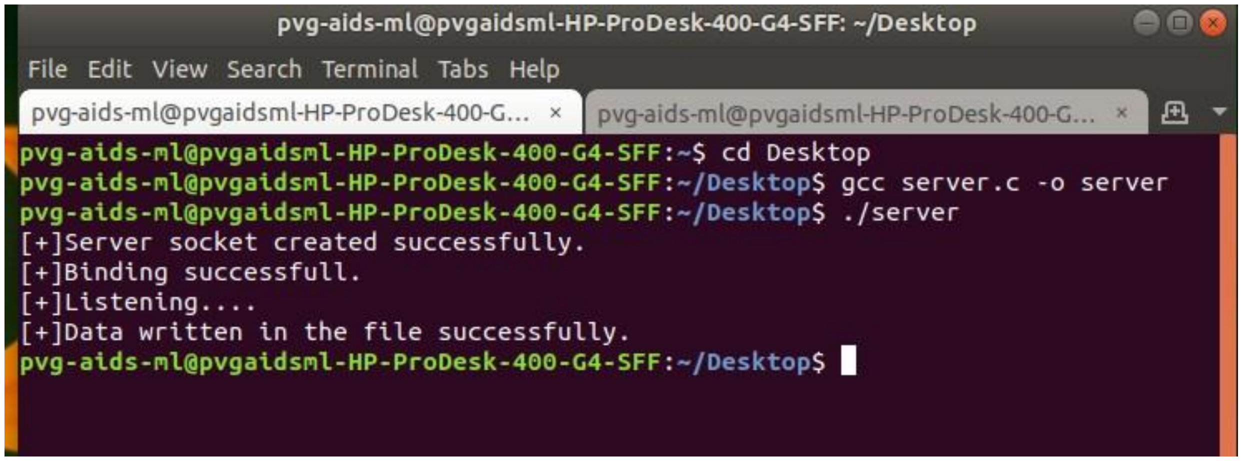
Program- UDP Server:

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

#define PORT 8080
#define BUFFER_SIZE 1024

void receive_file(int sockfd, struct sockaddr_in client_addr) {
    char buffer[BUFFER_SIZE] = {0};
    char *ack = "ACK";
    FILE *fp;
    socklen_t addr_len = sizeof(client_addr);
    ssize_t n;
    char *filename = "received_file";

    // Open file to write received data
    fp = fopen(filename, "wb");
    if (fp == NULL) {
        perror("File open error");
        exit(1);
    }

    while (1) {
        // Receive data from client
        n = recvfrom(sockfd, buffer, BUFFER_SIZE, 0, (struct sockaddr*)&client_addr, &addr_len);
        if (n <= 0) {
            break;
        }

        // Check if end of file is reached
        if (strcmp(buffer, "EOF") == 0) {
            printf("File transfer complete.\n");
            break;
        }

        // Write received data to file
        fwrite(buffer, sizeof(char), n, fp);
        bzero(buffer, BUFFER_SIZE);

        // Send acknowledgment to the client
        sendto(sockfd, ack, strlen(ack), 0, (struct sockaddr*)&client_addr, addr_len);
    }

    fclose(fp);
}

int main() {
    int sockfd;
    struct sockaddr_in server_addr, client_addr;

    // Create UDP socket
    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    // Initialize server address
    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(PORT);

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

    printf("Server is waiting for file...\n");

    // Receive file from client
    receive_file(sockfd, client_addr);

    // Close the socket
    close(sockfd);

    return 0;
}
```

Output: (Client - Server)


pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF: ~/Desktop
File Edit View Search Terminal Tabs Help
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G...
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G...

```
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF:~/Desktop$ gcc client.c -o client
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF:~/Desktop$ ./client
[+]Server socket created successfully.
[+]Connected to Server.
[+]File data sent successfully.
[+]Closing the connection.
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF:~/Desktop$ 
```


pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF: ~/Desktop
File Edit View Search Terminal Tabs Help
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G...
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G...

```
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF:~$ cd Desktop
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF:~/Desktop$ gcc server.c -o server
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF:~/Desktop$ ./server
[+]Server socket created successfully.
[+]Binding successfull.
[+]Listening....
[+]Data written in the file successfully.
pvg-aids-ml@pvgaidsml-HP-ProDesk-400-G4-SFF:~/Desktop$ 
```