EXPERIMENT: 26

TO IMPLEMENT DATE AND TIME DISPLAY FROM CLIENT TO SERVER USING TCP SOCKETS IN C

Aim: To implement date and time display from client to server using TCP Sockets.

Algorithm: Server
1. Create a server socket and bind it to port.

2. Listen for new connection and when a connection arrives, accept it.
3. Send server‟s date and time to the client.
4. Read client‟s IP address sent by the client.
5. Display the client details.
6. Repeat steps 2-5 until the server is terminated.
7. Close all streams.
8. Close the server socket.
9. Stop.

Client

1. Create a client socket and connect it to the server‟s port number.
2. Retrieve its own IP address using built-in function.
3. Send its address to the server.
4. Display the date & time sent by the server.
5. Close the input and output streams.
6. Close the client socket.
7. Stop.

Code:

Server Code (server.c)

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

#define PORT 8080

int main() {
    int server_fd, new_socket;
    struct sockaddr_in address;
```

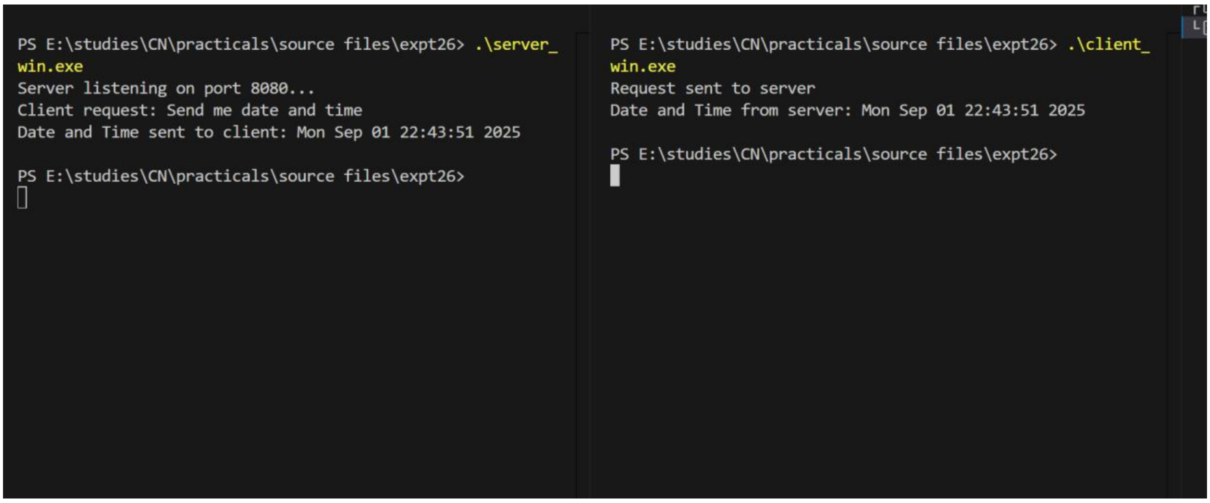
```
int addrlen = sizeof(address);
char buffer[1024] = {0};
time_t now;
char *dt;

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

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

if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {
   perror("Bind failed");
   exit(EXIT_FAILURE);
}

// Listen
if (listen(server_fd, 3) < 0) {
   perror("Listen failed");
   exit(EXIT_FAILURE);
}
printf("Server listening on port %d...\n", PORT);

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

// Read request from client
read(new_socket, buffer, 1024);
printf("Client request: %s\n", buffer);

// Get current date and time
now = time(NULL);
dt = ctime(&now);

// Send date and time to client
send(new_socket, dt, strlen(dt), 0);
printf("Date and Time sent to client: %s\n", dt);
```

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

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

#define PORT 8080

int main() {
    int sock = 0;
    struct sockaddr_in serv_addr;
    char buffer[1024] = {0};

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

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);

    // Convert IPv4/IPv6 address
    if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) <= 0) {
        printf("Invalid address/ Address not supported\n");
        return -1;
    }

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

    // Send request
    char *msg = "Send me date and time";
    send(sock, msg, strlen(msg), 0);
    printf("Request sent to server\n");

    // Receive date and time
    read(sock, buffer, 1024);
    printf("Date and Time from server: %s\n", buffer);

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

Commands to compile:
gcc server.c -o server_win -lws2_32
gcc client.c -o client_win -lws2_32

Commands to run:
First change you current dir to the dir where server.c and client.c are present.
Run it in 2 separate terminals

Server_win.exe
client_win.exe



```
PS E:\studies\CN\practicals\source files\expt26> .\server_
win.exe
Server listening on port 8080...
Client request: Send me date and time
Date and Time sent to client: Mon Sep 01 22:43:51 2025

PS E:\studies\CN\practicals\source files\expt26>
```

```
PS E:\studies\CN\practicals\source files\expt26> .\client_
win.exe
Request sent to server
Date and Time from server: Mon Sep 01 22:43:51 2025

PS E:\studies\CN\practicals\source files\expt26>
```

Result: Thus date and time is displayed from client to server using TCP Sockets successfully.