# ASSIGNMENT 9

## Question 1

Design a distributed application which consists of a server and client using threads.

### Code:

#### Server (Multithreaded Server using Sockets and Threads in C)

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <pthread.h>  
#include <unistd.h>  
#include <arpa/inet.h>  
  
#define PORT 8080  
#define BUFFER\_SIZE 1024  
  
void \*client\_handler(void \*socket\_desc) {  
 int client\_sock = \*(int \*)socket\_desc;  
 char buffer[BUFFER\_SIZE];  
  
 while (1) {  
 memset(buffer, 0, BUFFER\_SIZE);  
 int read\_size = read(client\_sock, buffer, BUFFER\_SIZE - 1);  
 if (read\_size <= 0) break;  
 printf("Client: %s", buffer);  
  
 send(client\_sock, buffer, strlen(buffer), 0);  
 }  
  
 close(client\_sock);  
 free(socket\_desc);  
 return NULL;  
}  
  
int main() {  
 int server\_sock, client\_sock, \*new\_sock;  
 struct sockaddr\_in server\_addr, client\_addr;  
 socklen\_t addr\_len = sizeof(struct sockaddr\_in);  
  
 server\_sock = socket(AF\_INET, SOCK\_STREAM, 0);  
 if (server\_sock == -1) {  
 perror("Socket creation failed");  
 exit(EXIT\_FAILURE);  
 }  
  
 server\_addr.sin\_family = AF\_INET;  
 server\_addr.sin\_addr.s\_addr = INADDR\_ANY;  
 server\_addr.sin\_port = htons(PORT);  
  
 if (bind(server\_sock, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) < 0) {  
 perror("Bind failed");  
 close(server\_sock);  
 exit(EXIT\_FAILURE);  
 }  
  
 listen(server\_sock, 3);  
 printf("Server listening on port %d...  
", PORT);  
  
 while ((client\_sock = accept(server\_sock, (struct sockaddr \*)&client\_addr, &addr\_len))) {  
 printf("Client connected.  
");  
  
 pthread\_t client\_thread;  
 new\_sock = malloc(1);  
 \*new\_sock = client\_sock;  
  
 if (pthread\_create(&client\_thread, NULL, client\_handler, (void \*)new\_sock) < 0) {  
 perror("Thread creation failed");  
 free(new\_sock);  
 close(client\_sock);  
 }  
 }  
  
 if (client\_sock < 0) {  
 perror("Accept failed");  
 close(server\_sock);  
 }  
  
 close(server\_sock);  
 return 0;  
}

#### Client

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <unistd.h>  
#include <arpa/inet.h>  
  
#define PORT 8080  
#define BUFFER\_SIZE 1024  
  
int main() {  
 int sock;  
 struct sockaddr\_in server\_addr;  
 char buffer[BUFFER\_SIZE];  
  
 sock = socket(AF\_INET, SOCK\_STREAM, 0);  
 if (sock == -1) {  
 perror("Socket creation failed");  
 exit(EXIT\_FAILURE);  
 }  
  
 server\_addr.sin\_family = AF\_INET;  
 server\_addr.sin\_port = htons(PORT);  
 server\_addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");  
  
 if (connect(sock, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) < 0) {  
 perror("Connection failed");  
 close(sock);  
 exit(EXIT\_FAILURE);  
 }  
  
 printf("Connected to the server. Type your messages:  
");  
  
 while (1) {  
 memset(buffer, 0, BUFFER\_SIZE);  
 fgets(buffer, BUFFER\_SIZE, stdin);  
 send(sock, buffer, strlen(buffer), 0);  
  
 memset(buffer, 0, BUFFER\_SIZE);  
 int read\_size = read(sock, buffer, BUFFER\_SIZE - 1);  
 if (read\_size <= 0) break;  
 printf("Server: %s", buffer);  
 }  
  
 close(sock);  
 return 0;  
}

### Sample Output:

Server Terminal:

Server listening on port 8080...  
Client connected.  
Client: Hello Server!  
Client: How are you?

Client Terminal:

Connected to the server. Type your messages:  
Hello Server!  
Server: Hello Server!  
How are you?  
Server: How are you?

## Question 2

Design a distributed application which consists of a stateless server using socket primitives.

### Code:

#### Stateless Server

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <unistd.h>  
#include <arpa/inet.h>  
  
#define PORT 8081  
#define BUFFER\_SIZE 1024  
  
int main() {  
 int server\_sock;  
 struct sockaddr\_in server\_addr, client\_addr;  
 socklen\_t addr\_len = sizeof(client\_addr);  
 char buffer[BUFFER\_SIZE];  
  
 server\_sock = socket(AF\_INET, SOCK\_DGRAM, 0);  
 if (server\_sock == -1) {  
 perror("Socket creation failed");  
 exit(EXIT\_FAILURE);  
 }  
  
 server\_addr.sin\_family = AF\_INET;  
 server\_addr.sin\_addr.s\_addr = INADDR\_ANY;  
 server\_addr.sin\_port = htons(PORT);  
  
 if (bind(server\_sock, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) < 0) {  
 perror("Bind failed");  
 close(server\_sock);  
 exit(EXIT\_FAILURE);  
 }  
  
 printf("Stateless server listening on port %d...  
", PORT);  
  
 while (1) {  
 memset(buffer, 0, BUFFER\_SIZE);  
 int len = recvfrom(server\_sock, buffer, BUFFER\_SIZE - 1, 0, (struct sockaddr \*)&client\_addr, &addr\_len);  
 if (len < 0) continue;  
  
 printf("Received: %s", buffer);  
  
 sendto(server\_sock, buffer, strlen(buffer), 0, (struct sockaddr \*)&client\_addr, addr\_len);  
 }  
  
 close(server\_sock);  
 return 0;  
}

#### Client

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <unistd.h>  
#include <arpa/inet.h>  
  
#define PORT 8081  
#define BUFFER\_SIZE 1024  
  
int main() {  
 int sock;  
 struct sockaddr\_in server\_addr;  
 char buffer[BUFFER\_SIZE];  
  
 sock = socket(AF\_INET, SOCK\_DGRAM, 0);  
 if (sock == -1) {  
 perror("Socket creation failed");  
 exit(EXIT\_FAILURE);  
 }  
  
 server\_addr.sin\_family = AF\_INET;  
 server\_addr.sin\_port = htons(PORT);  
 server\_addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");  
  
 printf("Stateless client. Type your messages:  
");  
  
 while (1) {  
 memset(buffer, 0, BUFFER\_SIZE);  
 fgets(buffer, BUFFER\_SIZE, stdin);  
 sendto(sock, buffer, strlen(buffer), 0, (struct sockaddr \*)&server\_addr, sizeof(server\_addr));  
  
 memset(buffer, 0, BUFFER\_SIZE);  
 int len = recvfrom(sock, buffer, BUFFER\_SIZE - 1, 0, NULL, NULL);  
 if (len < 0) continue;  
 printf("Server: %s", buffer);  
 }  
  
 close(sock);  
 return 0;  
}

### Sample Output:

Server Terminal:

Stateless server listening on port 8081...  
Received: Hello Stateless Server!  
Received: How's it going?

Client Terminal:

Stateless client. Type your messages:  
Hello Stateless Server!  
Server: Hello Stateless Server!  
How's it going?  
Server: How's it going?