ICS 2305 – SYSTEM PROGRAMMING ASSIGNMENT

NAME: PHILOMEN KHAMAL URENDI

REG No: SCT221-0799/2022

Creating a simple money transfer system using socket programming in C involves implementing both a client and a server component.

User Data Structure

```
#define MAX_USERS 100

#define USERNAME_LENGTH 20


typedef struct {

   char username[USERNAME_LENGTH];

   double balance;

} User;


User users[MAX_USERS];

int user_count = 0;
```

Server Code

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <arpa/inet.h>

#include <unistd.h>
```

```
#define PORT 8080

#define MAX_USERS 100

#define USERNAME_LENGTH 20


typedef struct {

   char username[USERNAME_LENGTH];

   double balance;

} User;


User users[MAX_USERS];

int user_count = 0;


void initialize_users() {

   // Initialize some users with balances

   strcpy(users[0].username, "Alice");

   users[0].balance = 1000.0;

   strcpy(users[1].username, "Bob");

   users[1].balance = 500.0;

   user_count = 2;

}


User* find_user(const char* username) {

   for (int i = 0; i < user_count; i++) {

      if (strcmp(users[i].username, username) == 0) {

         return &users[i];

      }
```

```
    }
    return NULL;
}
void handle_client(int client_socket) {
    char buffer[256];
    while (1) {
        bzero(buffer, sizeof(buffer));
        read(client_socket, buffer, sizeof(buffer));

        if (strcmp(buffer, "exit") == 0) {
            break;
        }
        char *sender = strtok(buffer, " ");
        char *recipient = strtok(NULL, " ");
        double amount = atof(strtok(NULL, " "));

        User* sender_user = find_user(sender);
        User* recipient_user = find_user(recipient);

        if (sender_user && recipient_user) {
            if (sender_user->balance >= amount) {
                sender_user->balance -= amount;
                recipient_user->balance += amount;
                sprintf(buffer, "Transfer successful! New balance: %s: %.2f, %s: %.2f\n",
                        sender_user->username, sender_user->balance,
                        recipient_user->username, recipient_user->balance);
            } else {
                sprintf(buffer, "Insufficient funds!\n");
            }
        } else {
            sprintf(buffer, "Invalid user(s)!\n");
        }
        write(client_socket, buffer, strlen(buffer));
    }
    close(client_socket);
}
int main() {
    int server_fd, client_socket;
    struct sockaddr_in address;
    int addrlen = sizeof(address);

    initialize_users();

    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 is listening on port %d\n", PORT);
```

```
   while ((client_socket = accept(server_fd, (struct sockaddr*)&address,
(socklen_t*)&addrlen))) {
      printf("Connection accepted\n");
      handle_client(client_socket);
   }
   close(server_fd);
   return 0;
}
```

Client Code

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

#define PORT 8080

int main() {
   int sock;
   struct sockaddr_in server_addr;
   char buffer[256];

   sock = socket(AF_INET, SOCK_STREAM, 0);
   server_addr.sin_family = AF_INET;
   server_addr.sin_port = htons(PORT);
```

```
    inet_pton(AF_INET, "127.0.0.1", &server_addr.sin_addr);

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

    while (1) {
        printf("Enter transaction (format: sender recipient amount) or 'exit' to quit: ");
        fgets(buffer, sizeof(buffer), stdin);
        buffer[strcspn(buffer, "\n")] = 0; // Remove newline

        write(sock, buffer, sizeof(buffer));

        if (strcmp(buffer, "exit") == 0) {
            break;
        }
        bzero(buffer, sizeof(buffer));
        read(sock, buffer, sizeof(buffer));
        printf("%s", buffer);
    }
    close(sock);
    return 0;
}
```