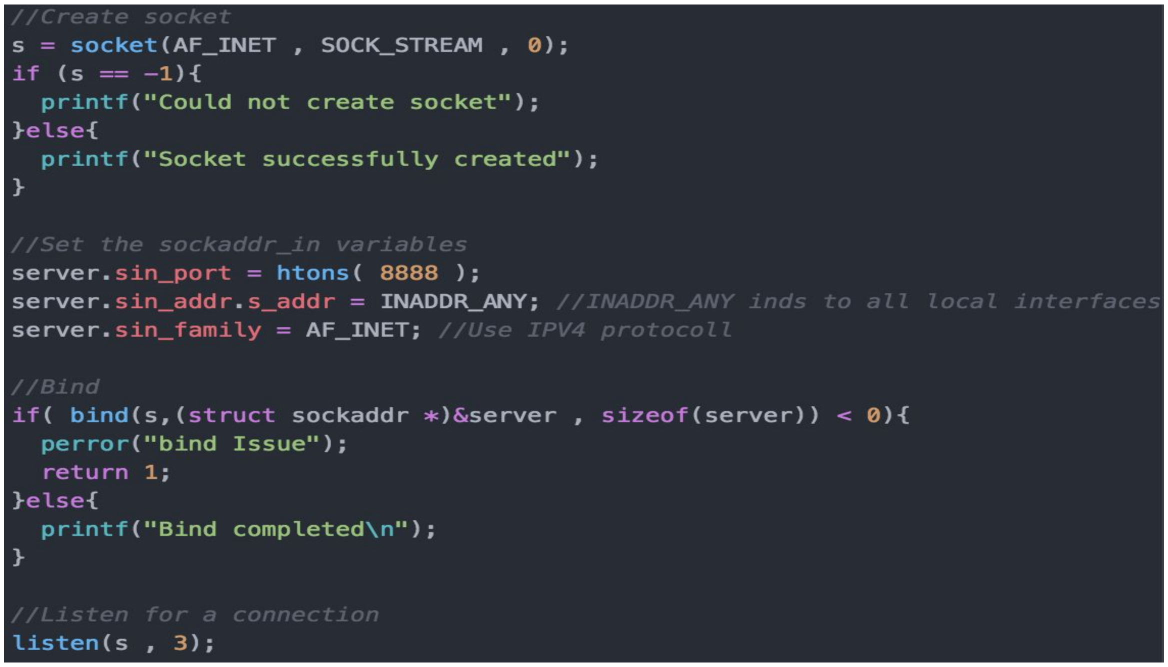
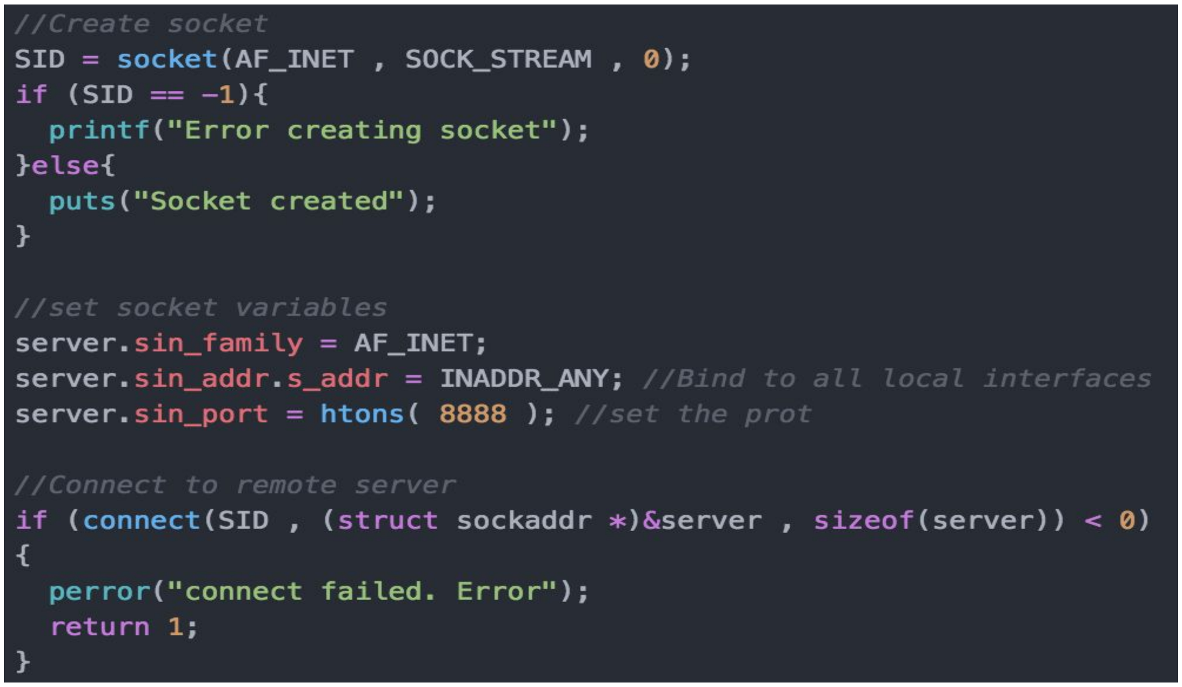
Systems Software Continuous Assessment 2
Owen Kane - C13383511

a.) Create a Server Socket program to run on the same server and the Intranet site



```
//Create socket
s = socket(AF_INET , SOCK_STREAM , 0);
if (s == -1){
  printf("Could not create socket");
}else{
  printf("Socket successfully created");
}

//Set the sockaddr_in variables
server.sin_port = htons( 8888 );
server.sin_addr.s_addr = INADDR_ANY; //INADDR_ANY inds to all local interfaces
server.sin_family = AF_INET; //Use IPV4 protocoll

//Bind
if( bind(s,(struct sockaddr *)&server , sizeof(server)) < 0){
  perror("bind Issue");
  return 1;
}else{
  printf("Bind completed\n");
}

//Listen for a connection
listen(s , 3);
```

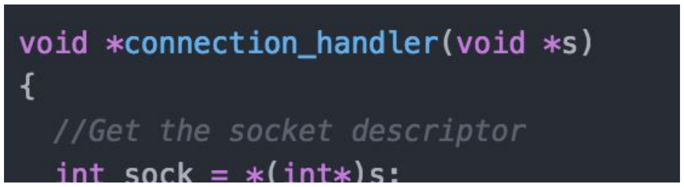
b) Create a Client program to connect to the server socket program.



```
//Create socket
SID = socket(AF_INET , SOCK_STREAM , 0);
if (SID == -1){
  printf("Error creating socket");
}else{
  puts("Socket created");
}

//set socket variables
server.sin_family = AF_INET;
server.sin_addr.s_addr = INADDR_ANY; //Bind to all local interfaces
server.sin_port = htons( 8888 ); //set the prot

//Connect to remote server
if (connect(SID , (struct sockaddr *)&server , sizeof(server)) < 0)
{
  perror("connect failed. Error");
  return 1;
}
```

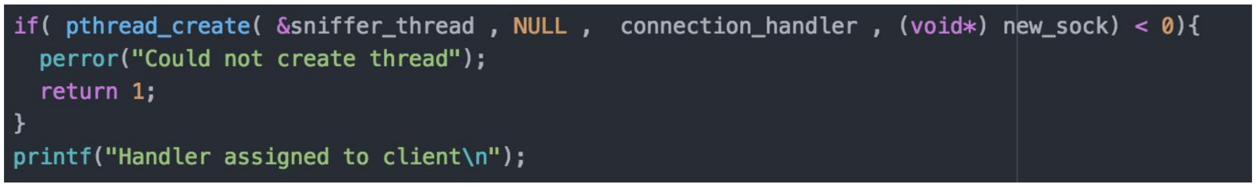
c) The system must be capable of handling multiple clients and transfers simultaneously.

I created a connection_handler to allow for multiple clients to interact with the system.



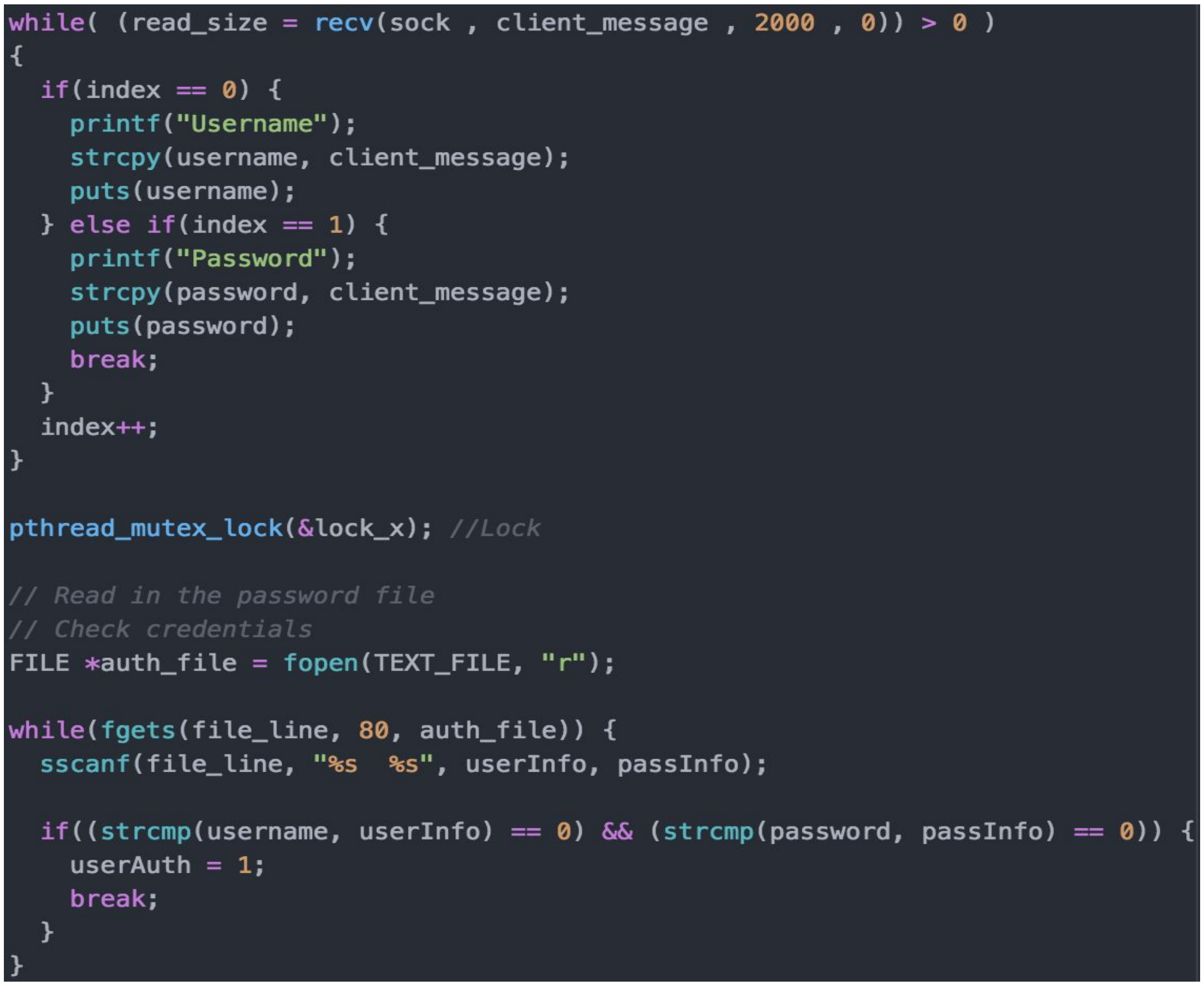
```
void *connection_handler(void *s)
{
  //Get the socket descriptor
  int sock = *(int*)s;
```

I then assign the client one of these handlers



```
if( pthread_create( &sniffer_thread , NULL ,  connection_handler , (void*) new_sock) < 0){
  perror("Could not create thread");
  return 1;
}
printf("Handler assigned to client\n");
```

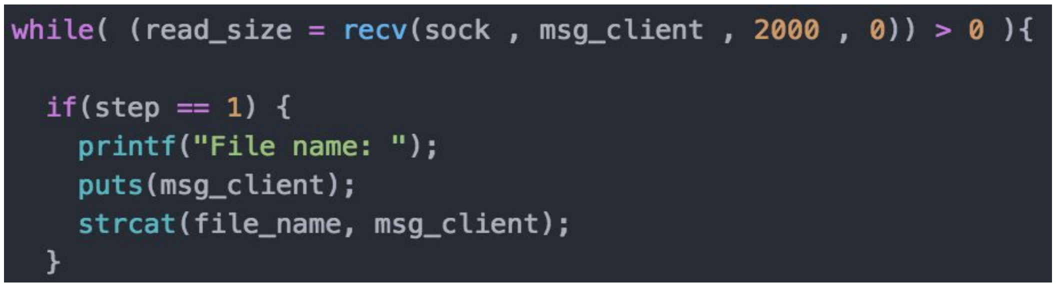
d) Clients must authenticate with the server program before any transfers are permitted



```
while( (read_size = recv(sock , client_message , 2000 , 0)) > 0 )
{
  if(index == 0) {
    printf("Username");
    strcpy(username, client_message);
    puts(username);
  } else if(index == 1) {
    printf("Password");
    strcpy(password, client_message);
    puts(password);
    break;
  }
  index++;
}

pthread_mutex_lock(&lock_x); //Lock

// Read in the password file
// Check credentials
FILE *auth_file = fopen(TEXT_FILE, "r");

while(fgets(file_line, 80, auth_file)) {
  sscanf(file_line, "%s  %s", userInfo, passInfo);

  if((strcmp(username, userInfo) == 0) && (strcmp(password, passInfo) == 0)) {
    userAuth = 1;
    break;
  }
}
```

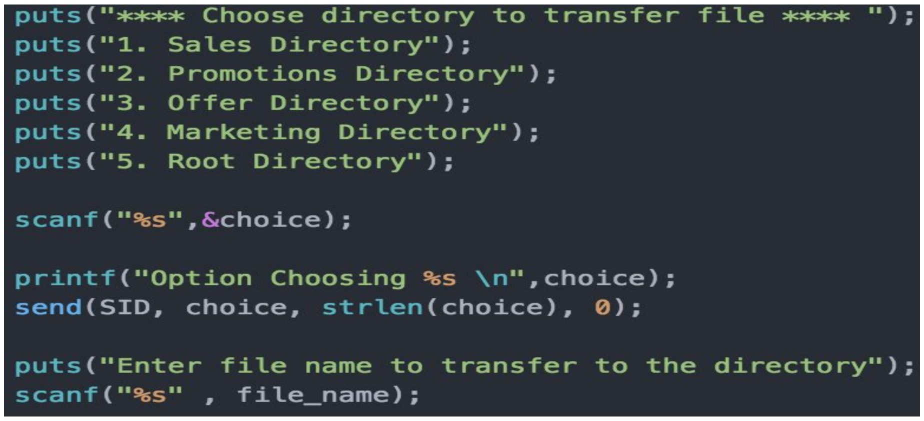
d) The client must take a filename and path via console and transfer this to the server to be stored. The following directories are where files can be transferred to:

Read in filename and path:



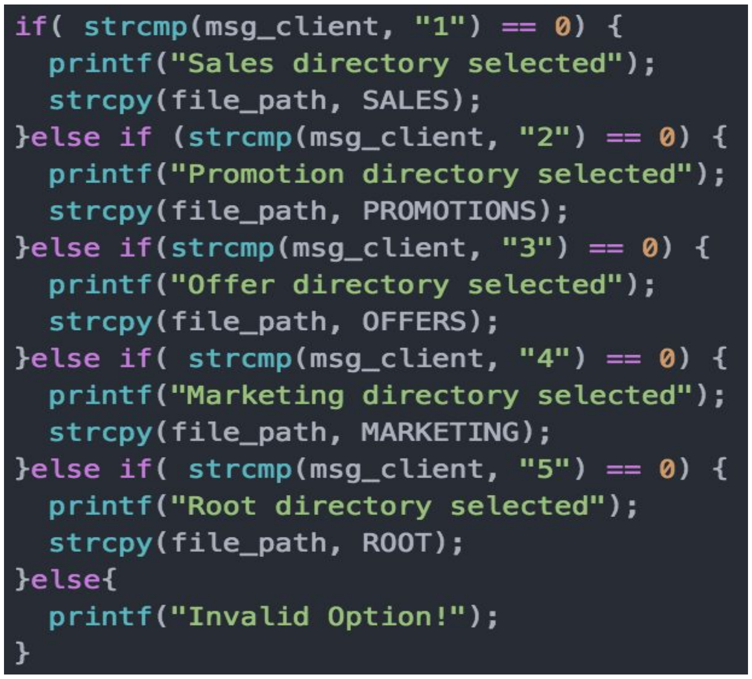
```
while( (read_size = recv(sock , msg_client , 2000 , 0)) > 0 ){

  if(step == 1) {
    printf("File name: ");
    puts(msg_client);
    strcat(file_name, msg_client);
  }
```

(Client)Select directory to save to on the server:



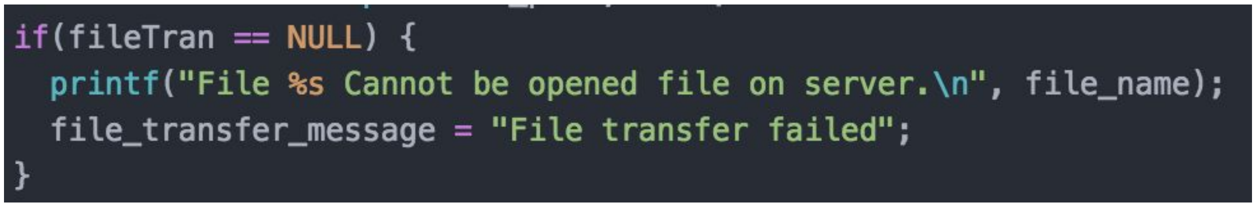
```
puts("**** Choose directory to transfer file **** ");
puts("1. Sales Directory");
puts("2. Promotions Directory");
puts("3. Offer Directory");
puts("4. Marketing Directory");
puts("5. Root Directory");

scanf("%s",&choice);

printf("Option Choosing %s \n",choice);
send(SID, choice, strlen(choice), 0);

puts("Enter file name to transfer to the directory");
scanf("%s" , file_name);
```

(Server) Take clients input and save file to that directory.



```
if( strcmp(msg_client, "1") == 0) {
  printf("Sales directory selected");
  strcpy(file_path, SALES);
}else if (strcmp(msg_client, "2") == 0) {
  printf("Promotion directory selected");
  strcpy(file_path, PROMOTIONS);
}else if(strcmp(msg_client, "3") == 0) {
  printf("Offer directory selected");
  strcpy(file_path, OFFERS);
}else if( strcmp(msg_client, "4") == 0) {
  printf("Marketing directory selected");
  strcpy(file_path, MARKETING);
}else if( strcmp(msg_client, "5") == 0) {
  printf("Root directory selected");
  strcpy(file_path, ROOT);
}else{
  printf("Invalid Option!");
}
```

e) The server must inform the client if the transfer was successful or not.



```
if(fileTran == NULL) {
  printf("File %s Cannot be opened file on server.\n", file_name);
  file_transfer_message = "File transfer failed";
}
```