**ШИНЖЛЭХ УХААН ТЕХНОЛОГИЙН ИХ СУРГУУЛЬ**

**Мэдээлэл холбооны технологийн сургууль**

****

**ЛАБОРАТОРИЙН АЖЛЫН**

**ТАЙЛАН - 4**

**Сүлжээний програмчлал I (F.NS351)**

**2022-2023 оны хичээлийн жилийн хавар**

**Лабораторийн ажлын нэр: Си програмчлалын хэлийг судлах-2**

**Хичээл заасан багш:** Б.Мөнхбаяр /Ph.D/

**Лабораторийн ажил гүйцэтгэсэн:** Уламбаяр /В200970314/

**Лабораторийн цаг:** 4-2

Улаанбаатар хот

2023 он

**Гарчиг**

Лабораторийн ажлын зорилго............................................................................3

Даалгавар….........................................................................................................4

Дүгнэлт................................................................................................................9

Ашигласан материал..........................................................................................9

**ЛАБОРАТОРИЙН АЖЛЫН ЗОРИЛГО, ЗОРИЛТ**

Энэхүү лабораторийн зорилго нь дараах шаардлагатай мэдлэг чадварыг олж авах явдал юм. Үүнд:

* socket programming
* client and server programming

**ЛАБОРАТОРИЙН АЖИЛ**

Даалгавар-2: Дараах даалгавруудыг гүйцэтгэнэ үү.

1. Клиент (tcp-socket-client.c) болон сервер (tcp-socket-server.c)-ийн эх кодыг

компайлдаад ажиллуулж үзнэ. Жич: Энэ бол жишээ код. Өөрсдөө эхнээс нь бас

өөрсдийнхөөрөө бичиж болно.

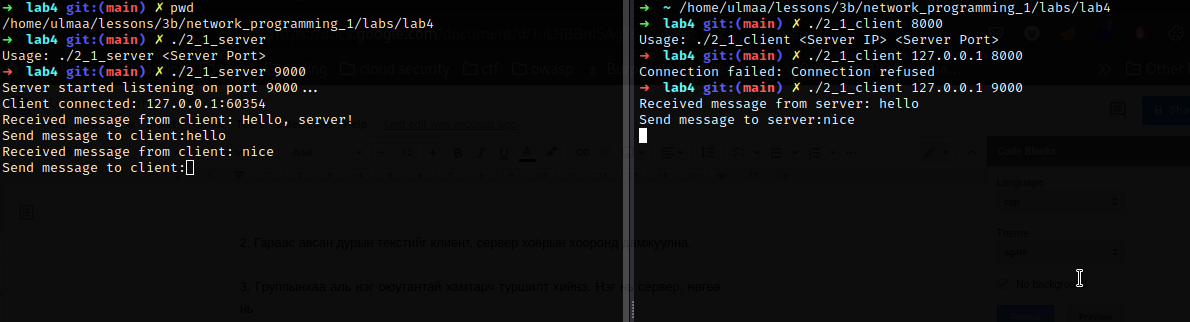
Tcp-socket-client.c

| #include <stdio.h> #include <stdlib.h> #include <string.h> #include <unistd.h> #include <sys/socket.h> #include <arpa/inet.h>  int main(int argc, char \*argv[]) {  // Check for correct number of arguments  if (argc != 3) {  printf("Usage: %s <Server IP> <Server Port>\n", argv[0]);  exit(1);  }   // Create a socket  int sock = socket(AF\_INET, SOCK\_STREAM, 0);  if (sock == -1) {  perror("Socket creation failed");  exit(1);  }   // Set up the server address  struct sockaddr\_in server\_addr;  memset(&server\_addr, 0, sizeof(server\_addr));  server\_addr.sin\_family = AF\_INET;  server\_addr.sin\_addr.s\_addr = inet\_addr(argv[1]);  server\_addr.sin\_port = htons(atoi(argv[2]));   // Connect to the server  if (connect(sock, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) == -1) {  perror("Connection failed");  exit(1);  }   // Send a message to the server  char message[1024] = "Hello, server!";  if (send(sock, message, strlen(message), 0) == -1) {  perror("Send failed");  exit(1);  }   // Receive a message from the server  while (1) {  char buffer[1024];  int num\_bytes = recv(sock, buffer, sizeof(buffer), 0);  if (num\_bytes == -1) {  perror("Receive failed");  exit(1);  }   // Print the message received from the server  buffer[num\_bytes] = '\0';  printf("Received message from server: %s\n", buffer);  printf("Send message to server:");  char message[1024];  scanf("%s",&message);  if (send(sock, message, strlen(message), 0) == -1) {  perror("Send failed");  exit(1);  }  // Close the socket    }  close(sock);  return 0; } |
| --- |

Tcp-socket-server.c

| #include <stdio.h> #include <stdlib.h> #include <string.h> #include <unistd.h> #include <sys/socket.h> #include <arpa/inet.h>  #define MAX\_CLIENTS 10  int main(int argc, char \*argv[]) {  // Check for correct number of arguments  if (argc != 2) {  printf("Usage: %s <Server Port>\n", argv[0]);  exit(1);  }   // Create a socket  int server\_sock = socket(AF\_INET, SOCK\_STREAM, 0);  if (server\_sock == -1) {  perror("Socket creation failed");  exit(1);  }   // Set up the server address  struct sockaddr\_in server\_addr;  memset(&server\_addr, 0, sizeof(server\_addr));  server\_addr.sin\_family = AF\_INET;  server\_addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);  server\_addr.sin\_port = htons(atoi(argv[1]));   // Bind the socket to the server address  if (bind(server\_sock, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) == -1) {  perror("Bind failed");  exit(1);  }   // Listen for incoming connections  if (listen(server\_sock, MAX\_CLIENTS) == -1) {  perror("Listen failed");  exit(1);  }   printf("Server started listening on port %s...\n", argv[1]);   // Accept incoming connections  struct sockaddr\_in client\_addr;  socklen\_t client\_addr\_len = sizeof(client\_addr);  int client\_sock;  char buffer[1024];  client\_sock = accept(server\_sock, (struct sockaddr \*)&client\_addr, &client\_addr\_len);  if (client\_sock == -1) {  perror("Accept failed");  exit(1);  }  printf("Client connected: %s:%d\n", inet\_ntoa(client\_addr.sin\_addr), ntohs(client\_addr.sin\_port));  while (1) {  // Receive a message from the client  int num\_bytes = recv(client\_sock, buffer, sizeof(buffer), 0);  if (num\_bytes == -1) {  perror("Receive failed");  exit(1);  }   // Print the message received from the client  buffer[num\_bytes] = '\0';  printf("Received message from client: %s\n");  printf("Send message to client: \n");  char message[1024];  scanf("%s",&message);  if (send(client\_sock, message, strlen(message), 0) == -1) {  perror("Send failed");  exit(1);  }  }  } |
| --- |

2. Гараас авсан дурын текстийг клиент, сервер хоёрын хооронд дамжуулна.

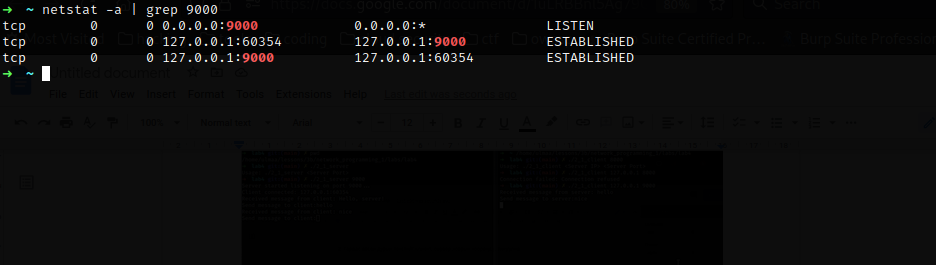


3. Группынхаа аль нэг оюутантай хамтарч туршилт хийнэ. Нэг нь сервер, нөгөө нь

клиент болоод хоорондоо текст дамжуулах буюу чатлана.

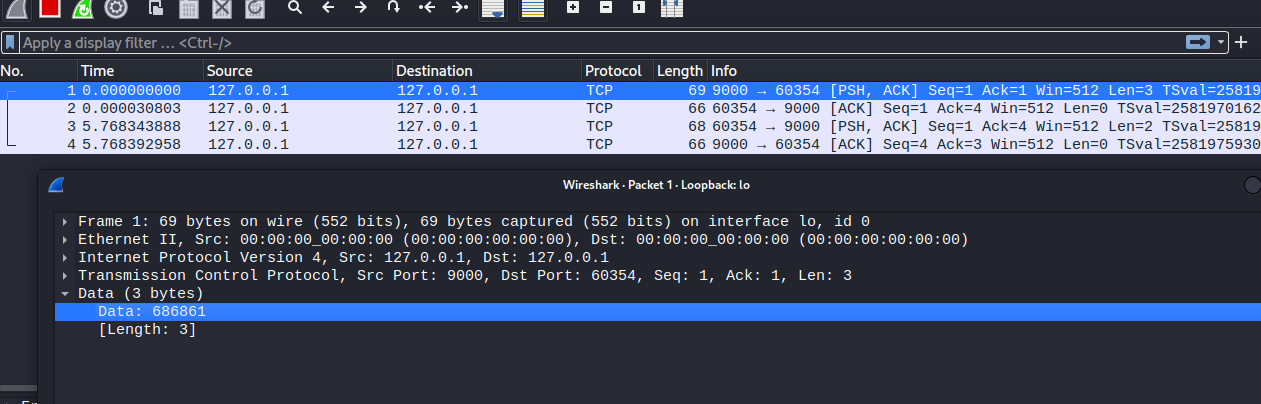
4. netstat командаар TCP холболт үүссэн эсэхийг шалгаж үзнэ. (Жишээ команд:

netstat -a | grep 8087).



5. Wireshark програмаар клиент, сервер хоёрын хооронд өгөгдөл дамжиж байгаа

эсэхийг шалгаж үзнэ.

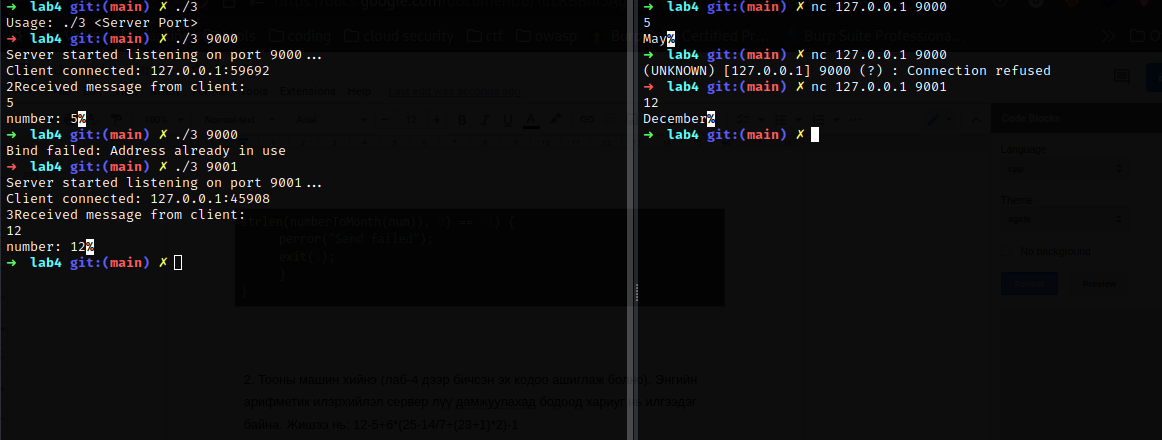


Даалгавар-3: Дараах даалгавруудыг гүйцэтгэнэ үү.

1. Клиент нь сервер лүү 1-12 хүртлэх тооны аль нэгийг дамжуулахад тухайн

тоотой тэнцүү сарын нэрийг сервер нь клиент рүү буцаадаг байна.

| #include <stdio.h> #include <stdlib.h> #include <string.h> #include <unistd.h> #include <sys/socket.h> #include <arpa/inet.h>  #define MAX\_CLIENTS 10 char\* numberToMonth(int month) {  char\* months[] = {  "January", "February", "March", "April", "May", "June",  "July", "August", "September", "October", "November", "December"  };   if (month < 1 || month > 12) {  return "Invalid month";  }   return months[month-1]; } int main(int argc, char \*argv[]) {  // Check for correct number of arguments  if (argc != 2) {  printf("Usage: %s <Server Port>\n", argv[0]);  exit(1);  }   // Create a socket  int server\_sock = socket(AF\_INET, SOCK\_STREAM, 0);  if (server\_sock == -1) {  perror("Socket creation failed");  exit(1);  }   // Set up the server address  struct sockaddr\_in server\_addr;  memset(&server\_addr, 0, sizeof(server\_addr));  server\_addr.sin\_family = AF\_INET;  server\_addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);  server\_addr.sin\_port = htons(atoi(argv[1]));   // Bind the socket to the server address  if (bind(server\_sock, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) == -1) {  perror("Bind failed");  exit(1);  }   // Listen for incoming connections  if (listen(server\_sock, MAX\_CLIENTS) == -1) {  perror("Listen failed");  exit(1);  }   printf("Server started listening on port %s...\n", argv[1]);   // Accept incoming connections  struct sockaddr\_in client\_addr;  socklen\_t client\_addr\_len = sizeof(client\_addr);  int client\_sock;  char buffer[1024];  client\_sock = accept(server\_sock, (struct sockaddr \*)&client\_addr, &client\_addr\_len);  if (client\_sock == -1) {  perror("Accept failed");  exit(1);  }  printf("Client connected: %s:%d\n", inet\_ntoa(client\_addr.sin\_addr), ntohs(client\_addr.sin\_port));    // Receive a message from the client  int num\_bytes = recv(client\_sock, buffer, sizeof(buffer), 0);  if (num\_bytes == -1) {  perror("Receive failed");  exit(1);  }  // Print the message received from the client  buffer[num\_bytes] = '\0';  printf("Received message from client: %s\n");  printf("%s",buffer);  int num = strtol(buffer, NULL, 10);  printf("number: %d",num);  if (send(client\_sock, numberToMonth(num), strlen(numberToMonth(num)), 0) == -1) {  perror("Send failed");  exit(1);  } } |
| --- |

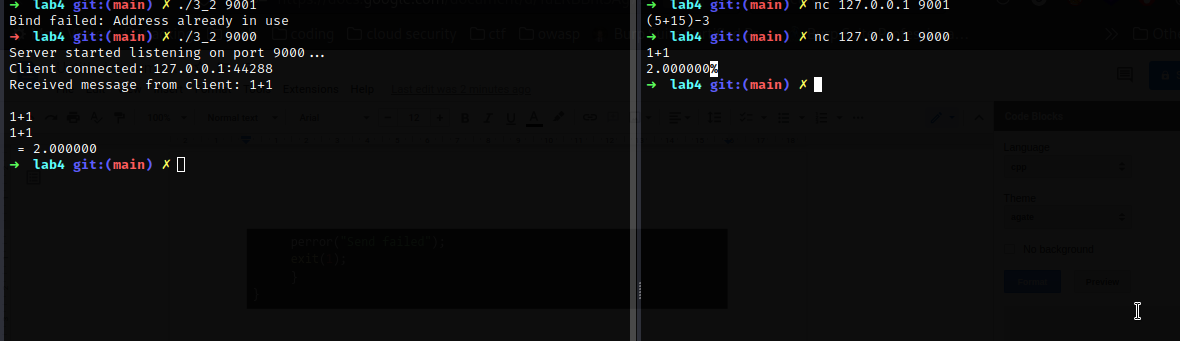


2. Тооны машин хийнэ (лаб-4 дээр бичсэн эх кодоо ашиглаж болно). Энгийн

арифметик илэрхийлэл сервер лүү дамжуулахад бодоод хариуг нь илгээдэг

байна. Жишээ нь: 12-5+6\*(25-14/7+(23+1)\*2)-1

| #include <stdio.h> #include <stdlib.h> #include <string.h> #include <unistd.h> #include <sys/socket.h> #include <arpa/inet.h> #include <math.h> float evaluate\_expression(char \*exp);  float parse\_number(char \*exp, int \*pos) {  float num = 0;  int decimal\_point = 0;   while (exp[\*pos] >= '0' && exp[\*pos] <= '9' || exp[\*pos] == '.') {  if (exp[\*pos] == '.') {  decimal\_point = 1;  } else {  num = num \* 10 + (exp[\*pos] - '0');  if (decimal\_point) {  decimal\_point \*= 10;  }  }  (\*pos)++;  }   if (decimal\_point) {  num /= decimal\_point;  }   return num; }  float parse\_factor(char \*exp, int \*pos) {  float result;  int is\_negative = 0;   if (exp[\*pos] == '-') {  is\_negative = 1;  (\*pos)++;  }   if (exp[\*pos] == '(') {  (\*pos)++;  result = evaluate\_expression(exp);  if (exp[\*pos] != ')') {  printf("Error: missing closing parenthesis\n");  return NAN;  }  (\*pos)++;  } else {  result = parse\_number(exp, pos);  }   if (is\_negative) {  result = -result;  }   return result; }  float parse\_term(char \*exp, int \*pos) {  float left, right;  char op;   left = parse\_factor(exp, pos);   while (exp[\*pos] == '\*' || exp[\*pos] == '/') {  op = exp[\*pos];  (\*pos)++;  right = parse\_factor(exp, pos);  if (op == '\*') {  left \*= right;  } else {  left /= right;  }  }   return left; }  float parse\_expression(char \*exp, int \*pos) {  float left, right;  char op;   left = parse\_term(exp, pos);   while (exp[\*pos] == '+' || exp[\*pos] == '-') {  op = exp[\*pos];  (\*pos)++;  right = parse\_term(exp, pos);  if (op == '+') {  left += right;  } else {  left -= right;  }  }   return left; }  float evaluate\_expression(char \*exp) {  int pos = 0;  return parse\_expression(exp, &pos); }   #define MAX\_CLIENTS 10 int main(int argc, char \*argv[]) {  // Check for correct number of arguments  if (argc != 2) {  printf("Usage: %s <Server Port>\n", argv[0]);  exit(1);  }   // Create a socket  int server\_sock = socket(AF\_INET, SOCK\_STREAM, 0);  if (server\_sock == -1) {  perror("Socket creation failed");  exit(1);  }   // Set up the server address  struct sockaddr\_in server\_addr;  memset(&server\_addr, 0, sizeof(server\_addr));  server\_addr.sin\_family = AF\_INET;  server\_addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);  server\_addr.sin\_port = htons(atoi(argv[1]));   // Bind the socket to the server address  if (bind(server\_sock, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) == -1) {  perror("Bind failed");  exit(1);  }   // Listen for incoming connections  if (listen(server\_sock, MAX\_CLIENTS) == -1) {  perror("Listen failed");  exit(1);  }   printf("Server started listening on port %s...\n", argv[1]);   // Accept incoming connections  struct sockaddr\_in client\_addr;  socklen\_t client\_addr\_len = sizeof(client\_addr);  int client\_sock;  char buffer[1024];  client\_sock = accept(server\_sock, (struct sockaddr \*)&client\_addr, &client\_addr\_len);  if (client\_sock == -1) {  perror("Accept failed");  exit(1);  }  printf("Client connected: %s:%d\n", inet\_ntoa(client\_addr.sin\_addr), ntohs(client\_addr.sin\_port));    // Receive a message from the client  char exp[] = "1+1";    int num\_bytes = recv(client\_sock, buffer, sizeof(buffer), 0);  if (num\_bytes == -1) {  perror("Receive failed");  exit(1);  }  // Print the message received from the client  buffer[num\_bytes] = '\0';   printf("Received message from client: %s\n");  printf("%s",buffer);  float result = evaluate\_expression(buffer);  printf("%s = %.6f\n", buffer, result);  sprintf(buffer, "%f", result);  if (send(client\_sock, buffer, strlen(buffer), 0) == -1) {  perror("Send failed");  exit(1);  } }#include <stdio.h> #include <stdlib.h> #include <string.h> #include <unistd.h> #include <sys/socket.h> #include <arpa/inet.h> #include <math.h> float evaluate\_expression(char \*exp);  float parse\_number(char \*exp, int \*pos) {  float num = 0;  int decimal\_point = 0;   while (exp[\*pos] >= '0' && exp[\*pos] <= '9' || exp[\*pos] == '.') {  if (exp[\*pos] == '.') {  decimal\_point = 1;  } else {  num = num \* 10 + (exp[\*pos] - '0');  if (decimal\_point) {  decimal\_point \*= 10;  }  }  (\*pos)++;  }   if (decimal\_point) {  num /= decimal\_point;  }   return num; }  float parse\_factor(char \*exp, int \*pos) {  float result;  int is\_negative = 0;   if (exp[\*pos] == '-') {  is\_negative = 1;  (\*pos)++;  }   if (exp[\*pos] == '(') {  (\*pos)++;  result = evaluate\_expression(exp);  if (exp[\*pos] != ')') {  printf("Error: missing closing parenthesis\n");  return NAN;  }  (\*pos)++;  } else {  result = parse\_number(exp, pos);  }   if (is\_negative) {  result = -result;  }   return result; }  float parse\_term(char \*exp, int \*pos) {  float left, right;  char op;   left = parse\_factor(exp, pos);   while (exp[\*pos] == '\*' || exp[\*pos] == '/') {  op = exp[\*pos];  (\*pos)++;  right = parse\_factor(exp, pos);  if (op == '\*') {  left \*= right;  } else {  left /= right;  }  }   return left; }  float parse\_expression(char \*exp, int \*pos) {  float left, right;  char op;   left = parse\_term(exp, pos);   while (exp[\*pos] == '+' || exp[\*pos] == '-') {  op = exp[\*pos];  (\*pos)++;  right = parse\_term(exp, pos);  if (op == '+') {  left += right;  } else {  left -= right;  }  }   return left; }  float evaluate\_expression(char \*exp) {  int pos = 0;  return parse\_expression(exp, &pos); }   #define MAX\_CLIENTS 10 int main(int argc, char \*argv[]) {  // Check for correct number of arguments  if (argc != 2) {  printf("Usage: %s <Server Port>\n", argv[0]);  exit(1);  }   // Create a socket  int server\_sock = socket(AF\_INET, SOCK\_STREAM, 0);  if (server\_sock == -1) {  perror("Socket creation failed");  exit(1);  }   // Set up the server address  struct sockaddr\_in server\_addr;  memset(&server\_addr, 0, sizeof(server\_addr));  server\_addr.sin\_family = AF\_INET;  server\_addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);  server\_addr.sin\_port = htons(atoi(argv[1]));   // Bind the socket to the server address  if (bind(server\_sock, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) == -1) {  perror("Bind failed");  exit(1);  }   // Listen for incoming connections  if (listen(server\_sock, MAX\_CLIENTS) == -1) {  perror("Listen failed");  exit(1);  }   printf("Server started listening on port %s...\n", argv[1]);   // Accept incoming connections  struct sockaddr\_in client\_addr;  socklen\_t client\_addr\_len = sizeof(client\_addr);  int client\_sock;  char buffer[1024];  client\_sock = accept(server\_sock, (struct sockaddr \*)&client\_addr, &client\_addr\_len);  if (client\_sock == -1) {  perror("Accept failed");  exit(1);  }  printf("Client connected: %s:%d\n", inet\_ntoa(client\_addr.sin\_addr), ntohs(client\_addr.sin\_port));    // Receive a message from the client  char exp[] = "1+1";    int num\_bytes = recv(client\_sock, buffer, sizeof(buffer), 0);  if (num\_bytes == -1) {  perror("Receive failed");  exit(1);  }  // Print the message received from the client  buffer[num\_bytes] = '\0';   printf("Received message from client: %s\n");  printf("%s",buffer);  float result = evaluate\_expression(buffer);  printf("%s = %.6f\n", buffer, result);  sprintf(buffer, "%f", result);  if (send(client\_sock, buffer, strlen(buffer), 0) == -1) {  perror("Send failed");  exit(1);  } } |
| --- |



Даалгавар-4: Клиент болон серверийн жишээ кодыг чатын програм болгох.

Одоогийн байдлаар клиент болон серверийн жишээ код нь сөөлжиж ажиллаж байгаа.

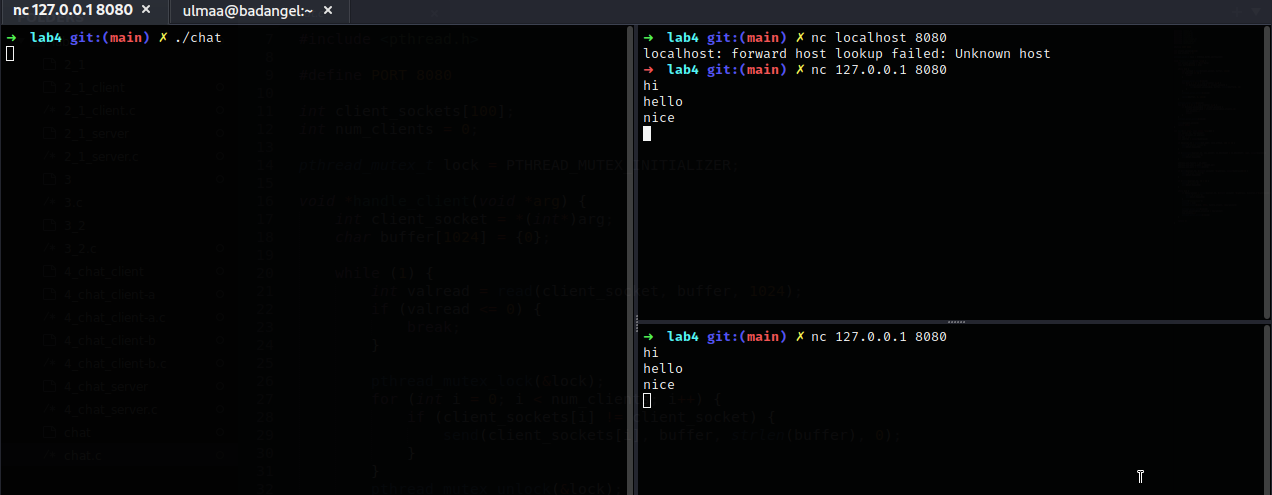
Ө.х нэг тал нь өгөгдлийг илгээхэд нөгөө тал нь хүлээж аваад хариуг өгнө. Хариуг нь

илгээгээгүй тохиолдолд дахиад өгөгдөл илгээж болохгүй байгаа.

1. Клиент, сервер хоёрыг чатлагч хоёр тал гэж үзээд аль нэг тал нь заавал хариуг

хүлээхгүйгээр дараалуулаад хэдэн ч мессежийг илгээж болдог байна

| #include <stdio.h> #include <stdlib.h> #include <string.h> #include <unistd.h> #include <sys/socket.h> #include <arpa/inet.h> #include <pthread.h>  #define PORT 8080  int client\_sockets[100]; int num\_clients = 0;  pthread\_mutex\_t lock = PTHREAD\_MUTEX\_INITIALIZER;  void \*handle\_client(void \*arg) {  int client\_socket = \*(int\*)arg;  char buffer[1024] = {0};   while (1) {  int valread = read(client\_socket, buffer, 1024);  if (valread <= 0) {  break;  }   pthread\_mutex\_lock(&lock);  for (int i = 0; i < num\_clients; i++) {  if (client\_sockets[i] != client\_socket) {  send(client\_sockets[i], buffer, strlen(buffer), 0);  }  }  pthread\_mutex\_unlock(&lock);   memset(buffer, 0, 1024);  }   pthread\_mutex\_lock(&lock);  for (int i = 0; i < num\_clients; i++) {  if (client\_sockets[i] == client\_socket) {  client\_sockets[i] = client\_sockets[num\_clients-1];  num\_clients--;  break;  }  }  pthread\_mutex\_unlock(&lock);   close(client\_socket);  return NULL; }  int main(int argc, char const \*argv[]) {  int server\_fd, new\_socket;  struct sockaddr\_in address;  int opt = 1;  int addrlen = sizeof(address);   if ((server\_fd = socket(AF\_INET, SOCK\_STREAM, 0)) == 0) {  perror("socket failed");  exit(EXIT\_FAILURE);  }   if (setsockopt(server\_fd, SOL\_SOCKET, SO\_REUSEADDR | SO\_REUSEPORT, &opt, sizeof(opt))) {  perror("setsockopt");  exit(EXIT\_FAILURE);  }   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);  }   if (listen(server\_fd, 3) < 0) {  perror("listen");  exit(EXIT\_FAILURE);  }   while (1) {  if ((new\_socket = accept(server\_fd, (struct sockaddr \*)&address, (socklen\_t\*)&addrlen))<0) {  perror("accept");  exit(EXIT\_FAILURE);  }  printf("Connection");  pthread\_t tid;  pthread\_create(&tid, NULL, handle\_client, &new\_socket);   pthread\_mutex\_lock(&lock);  client\_sockets[num\_clients] = new\_socket;  num\_clients++;  pthread\_mutex\_unlock(&lock);  }   return 0; } |
| --- |



**Дүгнэлт**

Энэхүү лабораторийн ажлаар “Сүлжээний програмчлал 1” хичээлийн суурь ойлголт болох socket programming талаар судалж мэдэж авсан ба өгөгдсөн бодлогуудын программыг бичэж гүйцэтгэсэн билээ.

**Ашигласан материал**

[1] “F.NS351 Сүлжээний програмчлал 4 хичээлийн лекцийн агуулга” 2022 он, Улаанбаатар хот, Монгол Улс

[2] “laboratory 4.pdf ” - Лабораторийн ажлын заавар

[3] https://www.tutorialspoint.com/cprogramming/

[4] https://www.tutorialspoint.com/cprogramming/c\_basic\_syntax.htm