Bài tập thực hành

Môn: TH Lập Trình Mạng

Chương 5: I/O Multiplexing

1. MỤC TIÊU

Bài thực hành đưa ra với mục đích giúp sinh viên hiểu biết về cơ chế I/O Multiplexing. Sinh viên sẽ có thể sử dụng một số hàm như select(), pselect(), poll() để cho tiến trình kiểm soát nhiều đầu vào cùng lúc (standard input, nhiều socket, ...). Cụ thể, bài thực hành này sẽ giúp sinh viên phát triển một ứng dụng chat thời gian thực, đồng thời hiểu sâu hơn về lập trình socket cơ bản và các kỹ thuật nâng cao trong việc xử lý nhiều client sử dụng I/O multiplexing.

2. YÊU CẦU

- Kiến thức cơ bản về lập trình C
- Kiến thức cơ bản về Mạng máy tính
- Máy tính cài đặt Hệ điều hành Linux (khuyến khích distro Ubuntu)
- Sử dụng các hàm trong bộ thư viện liên quan lập trình mạng

3. BÀI THỰC HÀNH

Phần 1: Lập Trình Socket TCP – Thiết Lập Cơ Bản

1.1: Triển Khai Server TCP

- Mục Tiêu: Triển khai một server TCP để quản lý phòng chat và xử lý nhiều kết nối client.
- Các Bước:
  1. Tạo socket server sử dụng `socket()`.

```
21      // Create a socket
22      server_socket = socket(AF_INET, SOCK_STREAM, 0);
23      if (server_socket < 0) {
24          perror("Socket failed");
25          exit(EXIT_FAILURE);
26      }
```

  2. Gán socket server tới một địa chỉ IP và cổng cụ thể bằng `bind()`.

```
33      // Bind the socket
34      if (bind(server_socket, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
35          perror("Bind failed");
36          close(server_socket);
37          exit(EXIT_FAILURE);
38      }
```

3. Sử dụng `listen()` để chấp nhận các kết nối client đến.

```
40      // Listen for incoming connections
41      if (listen(server_socket, 3) < 0) {
42          perror("Listen failed");
43          close(server_socket);
44          exit(EXIT_FAILURE);
45      }
```

4. Chấp nhận nhiều kết nối client sử dụng `accept()`.

```
74      // Handling new connections
75      if (FD_ISSET(server_socket, &readfds)) {
76          if ((client_socket = accept(server_socket, (struct sockaddr *)&client_addr, &addr_len)) < 0) {
77              perror("Accept failed");
78              exit(EXIT_FAILURE);
79          }
80
81          printf("\nNew connection, socket fd is %d, ip is: %s, port: %d\n",
82                 client_socket, inet_ntoa(client_addr.sin_addr), ntohs(client_addr.sin_port));
```

5. Phát các tin nhắn nhận được từ một client đến tất cả các client đang kết nối khác.

```
108                     // Broadcast message to all clients
109                     for (int j = 0; j < MAX_CLIENTS; j++) {
110                         if (client_sockets[j] != 0 && client_sockets[j] != sd) {
111                             send(client_sockets[j], buffer, strlen(buffer), 0);
112                         }
113                     }
```

2.2: Triển Khai Client TCP

- Mục Tiêu: Triển khai một client TCP kết nối tới server phòng chat và tương tác với các client khác thông qua server.
- Các Bước:
  1. Tạo socket client sử dụng socket().

```
17      // Create a socket
18      if ((client_socket = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
19          perror("Socket error");
20          return -1;
21      }
```

2. Kết nối tới server sử dụng connect().

```
33      if (connect(client_socket, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
34          perror("Connection failed");
35          return -1;
36      }
```

3. Gửi các tin nhắn do người dùng nhập tới server.

```
48              // Data from stdin
49              if (FD_ISSET(STDIN_FILENO, &readfds)) {
50                  fgets(buffer, BUFFER_SIZE, stdin);
51                  send(client_socket, buffer, strlen(buffer), 0);
52              }
```

4. Liên tục nhận và hiển thị các tin nhắn từ các client khác, được chuyển qua server.

```
54              // Message from server
55              if (FD_ISSET(client_socket, &readfds)) {
56                  int valread = read(client_socket, buffer, BUFFER_SIZE);
57                  buffer[valread] = '\0';
58                  printf("Message from server: %s\n", buffer);
59              }
```

Kết Quả Mong Đợi:

- Server có thể xử lý nhiều client, phát tin nhắn từ một client tới tất cả các client khác trong phòng chat.

```
phuc215118@sau:~/network$ ./server_select
Server listening on port 8080

New connection, socket fd is 4, ip is: 127.0.0.1, port: 48256
Adding client to list of sockets at index 1

New connection, socket fd is 5, ip is: 127.0.0.1, port: 51302
Adding client to list of sockets at index 2
```

- Mỗi client có thể gửi tin nhắn và nhìn thấy tin nhắn của các client khác trong thời gian thực.
  + *Client 1:*

```
phuc215118@sau:~/network$ ./client_select
Connected to the server...
client 1
Message from server: client 2
```

  + *Client 2:*

```
phuc215118@sau:~/network$ ./client_select
Connected to the server...
Message from server: client 1

client 2
```

Phần 2: Logic Server và Phát Tin Nhắn

2.1: Phát Tin Nhắn Đến Tất Cả Các Client

- Mục Tiêu: Triển khai logic chính của server để nhận tin nhắn từ một client và phát chúng đến tất cả các client khác đang kết nối.
- Các Bước:
  1. Khi server nhận được tin nhắn từ một client, nó sẽ chuyển tiếp tin nhắn đó tới tất cả các client khác.
  2. Sử dụng hàm send() để phát tin nhắn tới mỗi socket client.

```
107                 buffer[valread] = '\0';
108                 // Broadcast message to all clients
109                 for (int j = 0; j < MAX_CLIENTS; j++) {
110                     if (client_sockets[j] != 0 && client_sockets[j] != sd) {
111                         send(client_sockets[j], buffer, strlen(buffer), 0);
112                     }
113                 }
```

Nhiệm Vụ 2: Xử Lý Client Ngắt Kết Nối

- Mục Tiêu: Đảm bảo server có thể xử lý việc client ngắt kết nối một cách êm đẹp.

- Các Bước:
  1. Phát hiện khi một client ngắt kết nối và xóa kết nối của họ khỏi danh sách client hoạt động.

```
98                  // Check client disconnection
99                  if ((valread = read(sd, buffer, BUFFER_SIZE)) == 0) {
100                     getpeername(sd, (struct sockaddr *)&client_addr, &addr_len);
101                     printf("\nClient disconnected, ip %s, port %d\n",
102                           inet_ntoa(client_addr.sin_addr), ntohs(client_addr.sin_port));
103
104                     close(sd);
105                     client_sockets[i] = 0; // Remove socket from list
```

Kết quả:

```
phuc215118@sau:~/network$ ./server_select
Server listening on port 8080

New connection, socket fd is 4, ip is: 127.0.0.1, port: 48256
Adding client to list of sockets at index 1

New connection, socket fd is 5, ip is: 127.0.0.1, port: 51302
Adding client to list of sockets at index 2

Client disconnected, ip 127.0.0.1, port 48256

Client disconnected, ip 127.0.0.1, port 51302
```

  2. Thông báo tới các client còn lại rằng một người dùng đã rời khỏi phòng chat (tùy chọn).

Phần 3: I/O Multiplexing để Xử Lý Nhiều Client

3.1: Sử Dụng `select()` để Xử Lý Nhiều Client

- Mục Tiêu: Mục Tiêu: Chỉnh sửa server để xử lý nhiều client đồng thời sử dụng `select()` để thực hiện I/O multiplexing.
- Các Bước:
  1. Sử dụng select() để giám sát socket server cho các kết nối đến và socket client cho các tin nhắn đến.

```
55          // Initialize the file descriptor set
56          FD_ZERO(&readfds);
57          FD_SET(STDIN_FILENO, &readfds); // Monitor standard input
58          FD_SET(server_socket, &readfds);  // Monitor the server socket
59          max_sd = server_socket;
60
61          // Add all client sockets
62          for (int i = 0; i < MAX_CLIENTS; i++) {
63              int sd = client_sockets[i];
64              if (sd > 0) FD_SET(sd, &readfds);
65              if (sd > max_sd) max_sd = sd;
66          }
67
68          // Use select() to monitor sockets
69          activity = select(max_sd + 1, &readfds, NULL, NULL, NULL);
70          if ((activity < 0) && (errno != EINTR)) {
71              perror("Select error");
72          }
```

2. Khi một client gửi tin nhắn, server sẽ đọc nó và phát tới các client khác.

```
97          if (FD_ISSET(sd, &readfds)) {
98              // Check client disconnection
99              if ((valread = read(sd, buffer, BUFFER_SIZE)) == 0) {
100                 getpeername(sd, (struct sockaddr *)&client_addr, &addr_len);
101                 printf("\nClient disconnected, ip %s, port %d\n",
102                        inet_ntoa(client_addr.sin_addr), ntohs(client_addr.sin_port));
103
104                 close(sd);
105                 client_sockets[i] = 0; // Remove socket from list
106             } else {
107                 buffer[valread] = '\0';
108                 // Broadcast message to all clients
109                 for (int j = 0; j < MAX_CLIENTS; j++) {
110                     if (client_sockets[j] != 0 && client_sockets[j] != sd) {
111                         send(client_sockets[j], buffer, strlen(buffer), 0);
112                     }
113                 }
114             }
115         }
```

3. Xử lý các kết nối client mới và ngắt kết nối client.

```
84          // Add new client to list
85          for (int i = 0; i < MAX_CLIENTS; i++) {
86              if (client_sockets[i] == 0) {
87                  client_sockets[i] = client_socket;
88                  printf("Adding client to list of sockets at index %d\n", i + 1);
89                  break;
90              }
91          }
```

```
99              if ((valread = read(sd, buffer, BUFFER_SIZE)) == 0) {
100                 getpeername(sd, (struct sockaddr *)&client_addr, &addr_len);
101                 printf("\nClient disconnected, ip %s, port %d\n",
102                       inet_ntoa(client_addr.sin_addr), ntohs(client_addr.sin_port));
103
104                 close(sd);
105                 client_sockets[i] = 0; // Remove socket from list
```

3.2: Sử Dụng pselect() để Thực Hiện I/O Multiplexing An Toàn Với Tín Hiệu

- Mục Tiêu: Minh họa cách `pselect()` có thể được sử dụng để xử lý nhiều client trong khi xử lý tín hiệu an toàn.
- Các Bước:
  1. Thiết lập xử lý tín hiệu (ví dụ: SIGINT để kết thúc server) bằng `pselect()`.
  + *Thiết lập xử lý tín hiệu:*

```
14  volatile sig_atomic_t waiter = 0;
15
16  void sigint_handler(int signo) {
17      waiter = 1;
18  }
```

  + *Thiết lập cấu trúc xử lý tín hiệu:*

```
39      // Set up signal handling
40      struct sigaction sa;
41      sa.sa_handler = sigint_handler;
42      sa.sa_flags = 0;
43      sigemptyset(&sa.sa_mask);
44      if (sigaction(SIGINT, &sa, NULL) == -1) {
45          perror("sigaction");
46          exit(1);
47      }
```

  2. Đóng server an toàn khi nhận tín hiệu, đảm bảo rằng tất cả các client đều ngắt kết nối một cách đúng đắn.

```
158     // Close all sockets
159     for (int i = 0; i < MAX_CLIENTS; i++) {
160         if (client_sockets[i] > 0) {
161             close(client_sockets[i]);
162         }
163     }
164     close(server_socket);
165 
166     printf("\nServer shutting down.\n");
```

Kết quả:

```
Client disconnected, ip 127.0.0.1, port 59852
^C
Server shutting down.
```

3.3: Sử Dụng `poll()` để Xử Lý Nhiều Client

- Mục Tiêu: Chỉnh sửa server để sử dụng `poll()` trong việc quản lý nhiều kết nối client.
- Các Bước:
  1. Sử dụng `poll()` để giám sát socket server và các socket client.

```
64      // Add server socket to poll
65      fds[0].fd = server_socket;
66      fds[0].events = POLLIN;
67  
68      while(1) {
69          int activity = poll(fds, nfds, -1);
70  
71          if (activity < 0) {
72              perror("poll error");
73              exit(EXIT_FAILURE);
74          }
```

  2. Thêm và xóa socket client ra khỏi mảng `pollfd` một cách động khi các client kết nối và ngắt kết nối.

+ *Thêm socket:*

```
76      // Check for events on server socket (new connection)
77      if (fds[0].revents & POLLIN) {
78          if ((new_socket = accept(server_socket, (struct sockaddr *)&address, (socklen_t*)&addrlen)) < 0) {
79              perror("accept");
80              exit(EXIT_FAILURE);
81          }
82
83          printf("\nNew connection, socket fd is %d, ip is: %s, port: %d\n",
84                 new_socket, inet_ntoa(address.sin_addr), ntohs(address.sin_port));
85
86          // Add new socket to array of sockets
87          if (nfds < MAX_CLIENTS + 1) {
88              fds[nfds].fd = new_socket;
89              fds[nfds].events = POLLIN;
90              nfds++;
91          } else {
92              printf("Too many connections, connection rejected\n");
93              close(new_socket);
94          }
95      }
```

+ *Xóa socket:*

```
97      // Check for I/O operation on client sockets
98      for (int i = 1; i < nfds; i++) {
99          if (fds[i].revents & POLLIN) {
100             int valread;
101             if ((valread = read(fds[i].fd, buffer, BUFFER_SIZE)) == 0) {
102                 // Client disconnected
103                 getpeername(fds[i].fd, (struct sockaddr*)&address, (socklen_t*)&addrlen);
104                 printf("Client disconnected, ip %s, port %d\n",
105                        inet_ntoa(address.sin_addr), ntohs(address.sin_port));
106                 close(fds[i].fd);
107                 fds[i].fd = -1;
108             }
109             else {
110                 // Broadcast the message to all other clients
111                 buffer[valread] = '\0';
112                 broadcast_message(fds, fds[i].fd, buffer, nfds);
113             }
114         }
115     }
```

```
117         // Remove disconnected clients from the array
118         for (int i = 1; i < nfds; i++) {
119             if (fds[i].fd == -1) {
120                 for(int j = i; j < nfds - 1; j++) {
121                     fds[j] = fds[j+1];
122                 }
123                 nfds--;
124                 i--;
125             }
126         }
```

3. Đảm bảo server có thể xử lý các kết nối mới và tin nhắn của client một cách hiệu quả.

```
13  void broadcast_message(struct pollfd *fds, int sender_fd, char *message, int nfds) {
14      for (int i = 1; i < nfds; i++) {
15          if (fds[i].fd != sender_fd && fds[i].fd != -1) {
16              send(fds[i].fd, message, strlen(message), 0);
17          }
18      }
19  }
```

Ví Dụ: Hàm quảng bá tin nhắn đến tất cả client đang kết nối

```
void broadcast_message(int sender_fd, int *client_sockets, int
num_clients, char *message) {

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

        int client_fd = client_sockets[i];

        if (client_fd != sender_fd && client_fd > 0) {

            send(client_fd, message, strlen(message), 0);

        }

    }

}
```

Phần 4: Tính Năng Phòng Chat và Trải Nghiệm Người Dùng

4.1: Thêm Tên Người Dùng

- Mục Tiêu: Cho phép các client đặt tên người dùng sẽ được hiển thị cùng với tin nhắn của họ.
- Các Bước:
  1. Khi kết nối, yêu cầu mỗi client cung cấp tên người dùng.
     + *Server:*

```
100             // Request username
101             char *welcome_msg = "Welcome! Please enter your username: ";
102             send(new_socket, welcome_msg, strlen(welcome_msg), 0);
```

```
135                 // Set username
136                 strncpy(clients[i].username, buffer, USERNAME_SIZE - 1);
137                 clients[i].username[strcspn(clients[i].username, "\n")] = 0; // Remove newline
```

+ *Client:*

```
63      // Receive and respond to username prompt
64      recv(client_socket, buffer, BUFFER_SIZE, 0);
65      printf("%s", buffer);
66      fgets(username, USERNAME_SIZE, stdin);
67      send(client_socket, username, strlen(username), 0);
```

2. Gán tên người dùng vào mỗi tin nhắn mà client gửi (ví dụ: `Alice: Hello!`).

Server:

```
144                 // Broadcast message
145                 char full_msg[BUFFER_SIZE];
146                 snprintf(full_msg, BUFFER_SIZE, "[%s]: %s", clients[i].username, buffer);
147                 broadcast_message(sd, full_msg);
```

4.2: Cải Thiện Giao Diện Người Dùng

- Mục Tiêu: Nâng cao giao diện cho client để hiển thị tin nhắn.
- Các Bước:
  1. Hiển thị rõ ràng các tin nhắn đến kèm theo tên người gửi.
     Client:

```
20      while ((read_size = recv(sock, buffer, BUFFER_SIZE, 0)) > 0) {
21          buffer[read_size] = '\0';
22          printf("%s", buffer);
23      }
```

  2. Liên tục yêu cầu người dùng nhập tin nhắn, trong khi cũng hiển thị các tin nhắn đến từ các client khác.
     Client:

```
69      // Create a thread to receive messages
70      if (pthread_create(&receive_thread, NULL, receive_messages, (void*)&client_socket) < 0) {
71          perror("Could not create thread");
72          return -1;
73      }
74
75      // Main loop to send messages
76      while (1) {
77          fgets(buffer, BUFFER_SIZE, stdin);
78          send(client_socket, buffer, strlen(buffer), 0);
79      }
```

4.3: Thông Báo Phòng Chat

- Mục Tiêu: Thông báo cho tất cả các client khi có người dùng mới tham gia hoặc rời khỏi phòng chat.
- Các Bước:
  1. Phát tin nhắn tới tất cả client khi có người dùng mới tham gia (ví dụ: `Alice đã tham gia phòng chat`).
     Server:

```
139                     // Notify other clients
140                     char join_msg[BUFFER_SIZE];
141                     snprintf(join_msg, BUFFER_SIZE, "[%s] has joined the chat.\n", clients[i].username);
142                     broadcast_message(sd, join_msg);
```

  2. Tương tự, thông báo khi một người dùng ngắt kết nối.
     Server:

```
123                 // Notify other clients
124                 char disconnect_msg[BUFFER_SIZE];
125                 snprintf(disconnect_msg, BUFFER_SIZE, "[%s] has left the chat.\n", clients[i].username);
126                 broadcast_message(sd, disconnect_msg);
127
```

Kết quả:

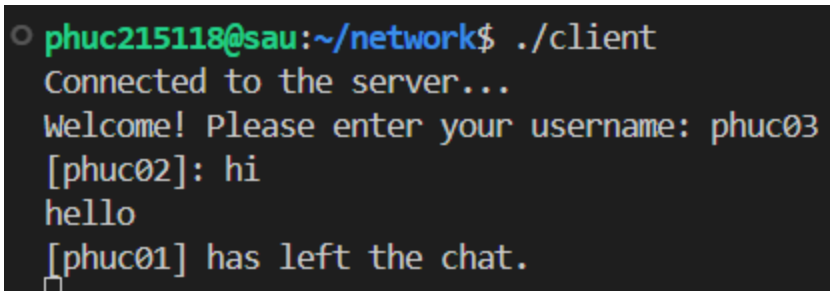
+ *Client 1:*

```
phuc215118@sau:~/network$ ./client
Connected to the server...
Welcome! Please enter your username: phuc01
[phuc02] has joined the chat.
[phuc03] has joined the chat.
[phuc02]: hi
[phuc03]: hello
^C
```

+ *Client 2:*

```
phuc215118@sau:~/network$ ./client
Connected to the server...
Welcome! Please enter your username: phuc02
[phuc03] has joined the chat.
hi
[phuc03]: hello
[phuc01] has left the chat.
```

+ *Client 3:*



Yêu Cầu Nộp Bài:

- Mã Nguồn: Nộp mã nguồn đầy đủ cho cả chương trình client và server.
- Trả lời trên Assignment: Viết một đoạn giải thích:
  + Cách mà các cơ chế I/O multiplexing (`select()`, `pselect()`, `poll()`) được sử dụng để quản lý nhiều client.
  + Cách mà chức năng phòng chat (phát tin nhắn, tên người dùng, v.v.) được triển khai.
  + Những khó khăn gặp phải trong quá trình triển khai và cách khắc phục

1. *Cơ chế I/O multiplexing được sử dụng để quản lý nhiều client:*
   - ***select()**: Hàm **select()** giám sát nhiều file descriptor (bao gồm cả socket) để kiểm tra sự kiện xảy ra (như dữ liệu đến từ client, hoặc kết nối mới). Hàm này chặn (block) tiến trình cho đến khi có sự kiện xảy ra.*
   - *pselect(): Tương tự hàm **select()**, nhưng quản lý tín hiệu an toàn hơn. Khi sử dụng **pselect()**, có thể thiết lập một tập hợp tín hiệu được chặn trong khi đang chờ sự kiện từ các socket.*
   - *poll(): Tương tự hàm **select()**, nhưng cho phép giám sát một số lượng socket lớn hơn và có thể sử dụng cấu trúc linh hoạt hơn. Hàm **poll()** cho phép thêm và xóa socket dễ dàng trong quá trình server hoạt động.*
2. *Cách chức năng phòng chat được triển khai:*
   - *Khởi tạo danh sách client: Tạo một mảng gồm socket và tên người dùng.*
   - *Hàm **select()** để chờ và kiểm tra các socket có hoạt động (như có kết nối mới hoặc tin nhắn từ client).*
   - *Sử dụng hàm **accept()** để chấp nhận kết nối mới. Yêu cầu client nhập tên người dùng. Thêm client vào danh sách.*
   - *Kiểm tra các socket client đang hoạt động: Gửi thông báo cho tất cả client khi có client mới tham gia hoặc khi một client ngắt kết nối.*
     + *Nếu một client ngắt kết nối (sử dụng hàm **read()** để phát hiện), thông báo cho các client khác và xóa client khỏi danh sách.*
     + *Gửi tên người dùng tới tất cả client còn lại: sử dụng **snprintf()** để định*

*dạng thông điệp và gọi **broadcast_message()** với thông điệp đó.*

3. *Khó khăn và cách khắc phục:*
   - *Khó khăn:*
     + *Chưa nắm vững biết cách thiết lập và sử dụng I/O multiplexing: **select()**, **pselect()**, **poll()**.*
     + *Xử lý khi client kết nối, ngắt kết nối.*
     + *Việc gửi nhận tin nhắn giữa các client.*
   - *Khắc phục:*
     + *Tìm hiểu và thực hành các hàm I/O multiplexing thông qua ví dụ cụ thể.*
     + *Sử dụng hàm **read()** để kiểm tra và quản lý danh sách client.*
     + *Sử dụng các hàm **send()** và **snprintf()** để định dạng và gửi tin nhắn, đồng thời đảm bảo rằng tất cả client đều nhận được tin nhăn đúng cách.*