Практикум на ЭВМ
client-server

Баев А.Ж.

Казахстанский филиал МГУ

23 ноября 2019

Семестровая работа.

1. Сервер
- общий процесс для подключения новых клиентов;
- индивидуальный процесс для связи с каждого клиента;
- общий процесс для логики.

2. Клиент
- один процесс для подключения к серверу;
- один процесс для обработки ввода.

include

```
1 #include <arpa/inet.h>
2 #include <sys/types.h>
3 #include <sys/socket.h>
4 #include <netdb.h>
5 #include <netinet/in.h>
6 #include <unistd.h>
```

Сервер. Создание сокета

```
1  //open socket, return socket descriptor
2  int server_socket = socket(PF_INET,
3                             SOCK_STREAM,
4                             0);
5
6  //set socket option
7  int socket_option = 1;
8  setsockopt(server_socket,
9             SOL_SOCKET,
10            SO_REUSEADDR,
11            &socket_option,
12            sizeof(socket_option));
```

Сервер. Привязка адреса

```
1  //set socket address
2  struct sockaddr_in server_address;
3  server_address.sin_family = AF_INET;
4  server_address.sin_port = htons(8080);
5  server_address.sin_addr.s_addr = INADDR_ANY;
6  bind(server_socket,
7       (struct sockaddr *) &server_address,
8       sizeof(server_address));
9
10 //start listen mode
11 listen(server_socket, 5);
```

Сервер. Подключения

```
1  while(1) {
2      struct sockaddr_in client;
3      struct sockaddr *client_ptr = &client;
4      socklen_t size;
5      int client_socket;
6      client_socket = accept(server_socket,
7                             client_ptr,
8                             &size);
9      char *addr = inet_ntoa(client.sin_addr);
10     int port = ntohs(client.sin_port);
11
12     printf("connected: %s %d\n", addr, port);
13
14     //write(client_socket, ...)
15
16     close(client_socket);
17 }
```

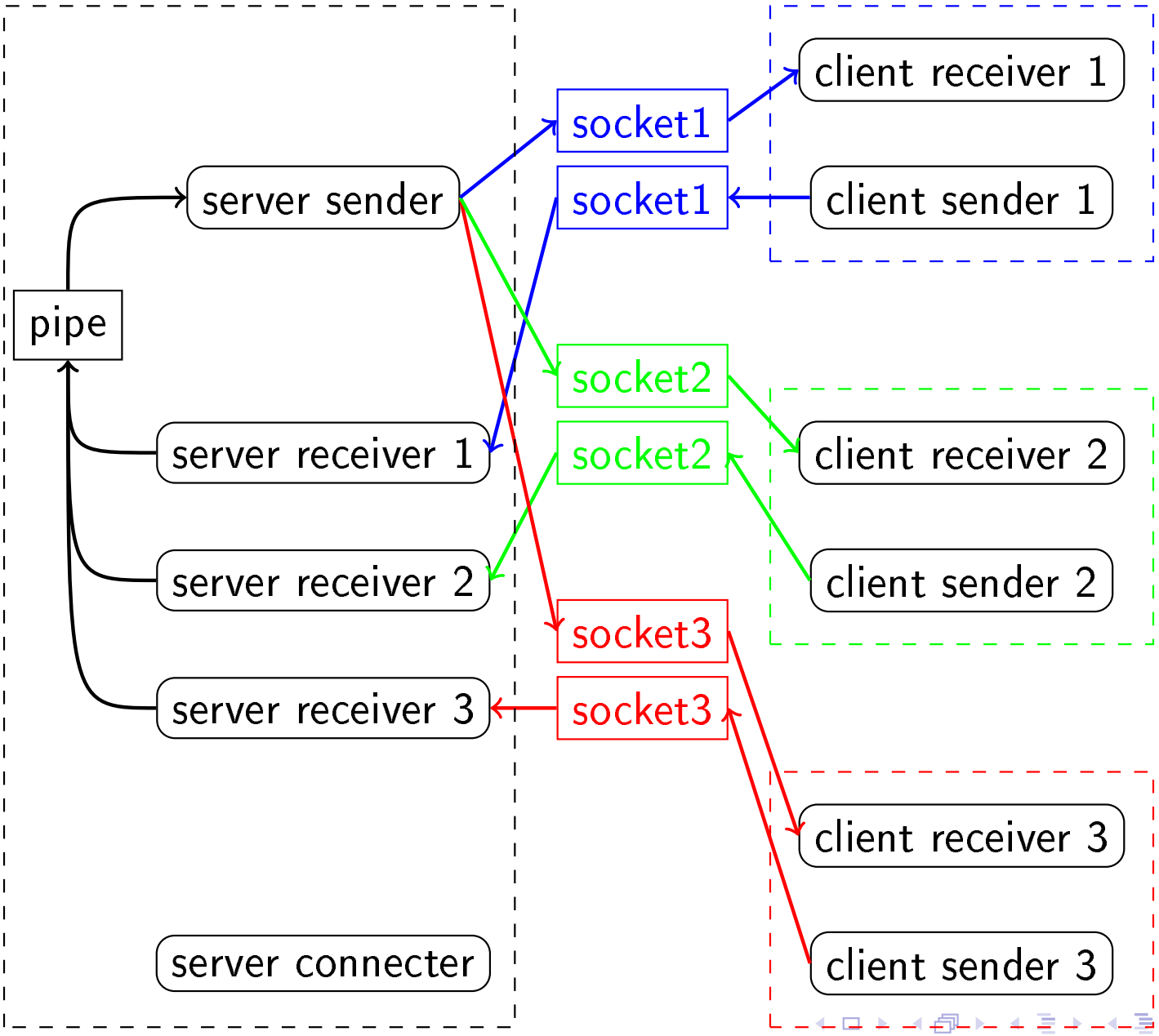
Клиент. Создание сокета

```
1  //open socket, return socket descriptor
2  int server_socket = socket(PF_INET,
3                             SOCK_STREAM,
4                             0);
5
6  //prepare server address
7  struct hostent *host = gethostbyname(ip);
8  struct sockaddr_in server_address;
9  server_address.sin_family = AF_INET;
10 server_address.sin_port = htons(port);
11 memcpy(&server_address.sin_addr,
12         host -> h_addr_list[0],
13         sizeof(server_address));
```

Клиент. Подключение

```
1  //connect
2  connect(server_socket,
3          (struct sockaddr *) &server_address,
4          sizeof(server_address));
5
6  //read(server_socket, ...)
```

Структура процессов



Формат передачи

sender	receiver	канал связи	формат
client	server	socket	size + data
server	server	pipe	idx + size + data
server	client	socket	idx + size + data

server connector

Подключение фиксированного количества пользователей.

```
1 for (i = 0; i < n; i++)
2   client_socket[i] = accept(...)
```

server receiver - создание

Инициализиация дочерних процессов для каждого пользователя.

```
 1 pipe(receiver_2_sender);
 2 for (i = 0; i < n; i++)
 3   if (fork() == 0) {
 4     close(receiver_2_sender[0]);
 5     reciever(receiver_2_sender[1],
 6              client_socket[i]);
 7     close(receiver_2_sender[1]);
 8     close(client_socket[i]);
 9     return 0;
10   }
11 close(receiver_2_sender[1]);
```

server receiver - пересылка в pipe

```
1  void reciever(int id, int input, int output) {
2    while (read(input, &size, sizeof(size)) > 0)
3      read(receiver_2_sender[0], buf, size);
4      /*critical start*/
5      write(output, &id, sizeof(id));
6      write(output, &size, sizeof(size));
7      write(output, buf, size);
8      /*critical stop*/
9    }
10 }
```

Комментариями выделена критическая секция (нужно добавить семафор)!

server sender

Получение сообщение от пользователя и рассылка остальным пользователям.

```
1  while(1) {
2    read(receiver_2_sender[0], &id, sizeof(id));
3    read(receiver_2_sender[0], &size, sizeof(size));
4    read(receiver_2_sender[0], buf, size);
5    for (i = 0; i < n; i++) {
6      write(client_socket[i], &id, sizeof(id));
7      write(client_socket[i], &size, sizeof(size));
8      write(client_socket[i], buf, size);
9    }
10 }
11 close(receiver_2_sender[0]);
```