



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет

*к лабораторной работе №6
«Сокеты»*

Студент: Батбилэг Н.

Группа: ИУ7-71Б

Оценка (баллы) _____

Преподаватель: Рязанова Н.Ю.

Москва.
2021 г.

1. Задание 1

- Написать приложение по модели клиент-сервер, демонстрирующее взаимодействие параллельных процессов на отдельном компьютере с использованием сокетов в файловом пространстве имен: семейство - AF_UNIX, тип - SOCK_DGRAM. При демонстрации работы программного комплекса необходимо запустить несколько клиентов (не меньше 5) и продемонстрировать, что сервер обрабатывает обращения каждого запущенного клиента.

В листингах 1 – 4 представлен код первого задания.

Листинг 1. Params.h.

```
1 #define MSG_SIZE 256
2 #define SOKET_NAME "mysocket.soc"
```

Листинг 2. Server.c.

```
1 #include <stdbool.h>
2 #include <stdio.h>
3 #include <sys/socket.h>
4 #include <signal.h>
5 #include <unistd.h>
6 #include "params.h"
7
8 #define SOCKET_ERR -1
9 #define BIND_ERR -2
10 #define RECV_ERR -3
11
12 bool sigint_flag = false;
13 int sockfd = 0;
14
15 void close_socket(const int sockfd, const char *socket_name) {
16     close(sockfd);
17     unlink(socket_name);
18 }
19
20 void sigint_handler(int signum) {
21     close_socket(sockfd, SOKET_NAME);
22     printf("Socket closed, signum = %d\n", signum);
23     sigint_flag = true;
24 }
25
26 int main() {
27     sockfd = socket(AF_LOCAL, SOCK_DGRAM, 0);
28     if (sockfd < 0) {
29         printf("Socket error\n");
30         return SOCKET_ERR;
31     }
32     struct sockaddr client_addr = {
33         .sa_family = AF_LOCAL,
34         .sa_data = SOKET_NAME
35     };
36     if (bind(sockfd, &client_addr, sizeof(client_addr)) < 0) {
37         close_socket(sockfd, SOKET_NAME);
38         printf("Bind error\n");
39         return BIND_ERR;
40     }
41
42     printf("Server %d started successfully\n", getpid());
```

```

43     signal(SIGINT, sigint_handler);
44
45     char msg[MSG_SIZE];
46     for (int i = 0;; i++) {
47         int sz = recv(sockfd, msg, sizeof(msg), 0);
48         if (sigint_flag) {
49             return 0;
50         }
51         else if (sz < 0) {
52             close_socket(sockfd, SOKET_NAME);
53             printf("Socket closed. RECV error\n");
54             return RECV_ERR;
55         }
56         msg[sz] = 0; //EOL
57         printf("Client %d sent: %s\n", i, msg);
58     }
59
60     close_socket(sockfd, SOKET_NAME);
61     printf("Socket closed.\n");
62     return 0;
63 }

```

Листинг 3. Client.c.

```

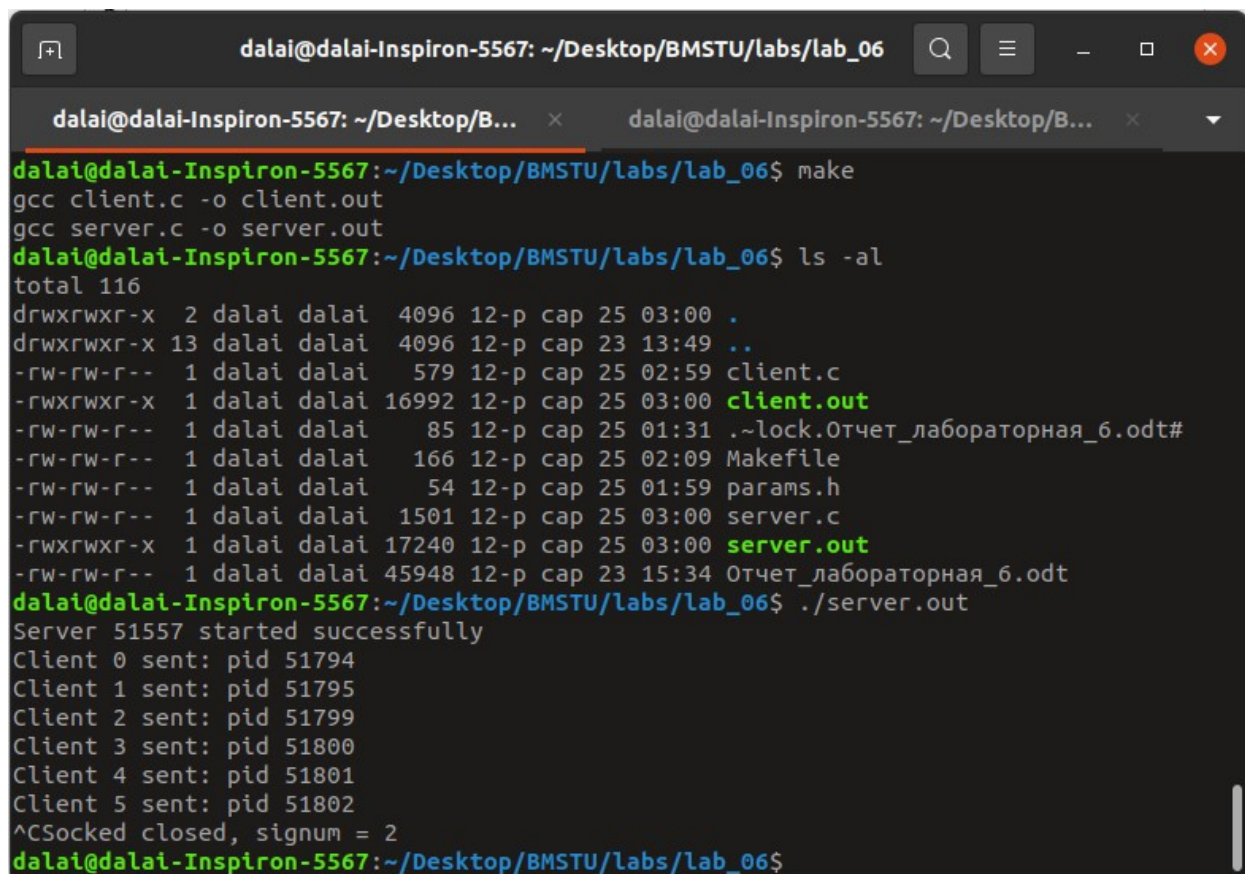
1  #include <sys/socket.h>
2  #include <stdio.h>
3  #include <unistd.h>
4  #include <string.h>
5  #include "params.h"
6
7  #define SOCKET_ERR -1
8
9  int main() {
10     int sockfd = socket(AF_LOCAL, SOCK_DGRAM, 0);
11     if (sockfd < 0) {
12         printf("Socket error\n");
13         return SOCKET_ERR;
14     }
15     struct sockaddr server_addr = {
16         .sa_family = AF_LOCAL,
17         .sa_data = SOKET_NAME
18     };
19     char msg[MSG_SIZE];
20     sprintf(msg, "pid %d", getpid());
21     sendto(sockfd, msg, strlen(msg), 0, &server_addr,
22         sizeof(server_addr));
23     printf("Sent message: %s\n", msg);
24     return 0;
25 }

```

Листинг 4. Makefile

```
1 all : client server
2
3 client : client.c params.h
4         gcc client.c -o client.out
5
6 server : server.c params.h
7         gcc server.c -o server.out
8
9 clean :
10        rm client.out server.out
```

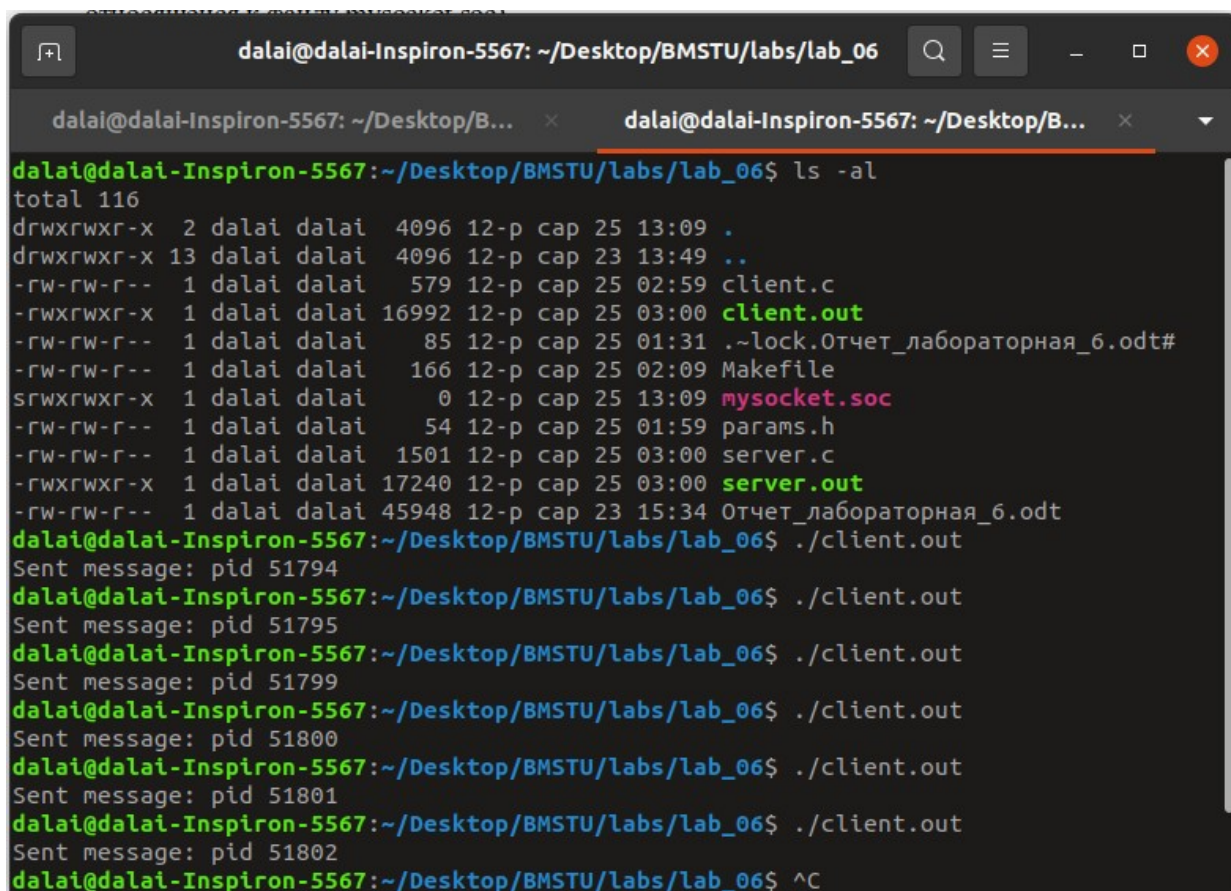
На рисунке 1 представлены сборка сервера и клиента, содержимое директории до запуска сервера и демонстрация работы сервера.



```
dalai@dalai-Inspiron-5567: ~/Desktop/BMSTU/labs/lab_06
dalai@dalai-Inspiron-5567: ~/Desktop/B... x dalai@dalai-Inspiron-5567: ~/Desktop/B... x
dalai@dalai-Inspiron-5567:~/Desktop/BMSTU/labs/lab_06$ make
gcc client.c -o client.out
gcc server.c -o server.out
dalai@dalai-Inspiron-5567:~/Desktop/BMSTU/labs/lab_06$ ls -al
total 116
drwxrwxr-x 2 dalai dalai 4096 12-p cap 25 03:00 .
drwxrwxr-x 13 dalai dalai 4096 12-p cap 23 13:49 ..
-rw-rw-r-- 1 dalai dalai 579 12-p cap 25 02:59 client.c
-rwxrwxr-x 1 dalai dalai 16992 12-p cap 25 03:00 client.out
-rw-rw-r-- 1 dalai dalai 85 12-p cap 25 01:31 ~/.lock.Отчет_лабораторная_6.odt#
-rw-rw-r-- 1 dalai dalai 166 12-p cap 25 02:09 Makefile
-rw-rw-r-- 1 dalai dalai 54 12-p cap 25 01:59 params.h
-rw-rw-r-- 1 dalai dalai 1501 12-p cap 25 03:00 server.c
-rwxrwxr-x 1 dalai dalai 17240 12-p cap 25 03:00 server.out
-rw-rw-r-- 1 dalai dalai 45948 12-p cap 23 15:34 Отчет_лабораторная_6.odt
dalai@dalai-Inspiron-5567:~/Desktop/BMSTU/labs/lab_06$ ./server.out
Server 51557 started successfully
Client 0 sent: pid 51794
Client 1 sent: pid 51795
Client 2 sent: pid 51799
Client 3 sent: pid 51800
Client 4 sent: pid 51801
Client 5 sent: pid 51802
^CSocket closed, signum = 2
dalai@dalai-Inspiron-5567:~/Desktop/BMSTU/labs/lab_06$
```

Рисунок 1. Результаты работы сервера.

На рисунке 2 представлены содержимое директории после запуска сервера и демонстрация работы клиента. Кроме того, на рисунке 2 присутствует подтверждение того, что сокет действительно создавался (это можно понять по первому символу строки s – сокет, относящейся к файлу mysocket.soc).



```
dalai@dalai-Inspiron-5567: ~/Desktop/BMSTU/labs/lab_06
dalai@dalai-Inspiron-5567: ~/Desktop/B... x dalai@dalai-Inspiron-5567: ~/Desktop/B... x
dalai@dalai-Inspiron-5567:~/Desktop/BMSTU/labs/lab_06$ ls -al
total 116
drwxrwxr-x 2 dalai dalai 4096 12-p cap 25 13:09 .
drwxrwxr-x 13 dalai dalai 4096 12-p cap 23 13:49 ..
-rw-rw-r-- 1 dalai dalai 579 12-p cap 25 02:59 client.c
-rwxrwxr-x 1 dalai dalai 16992 12-p cap 25 03:00 client.out
-rw-rw-r-- 1 dalai dalai 85 12-p cap 25 01:31 ~/.lock.Отчет_лабораторная_6.odt#
-rw-rw-r-- 1 dalai dalai 166 12-p cap 25 02:09 Makefile
srwxrwxr-x 1 dalai dalai 0 12-p cap 25 13:09 mysocket.soc
-rw-rw-r-- 1 dalai dalai 54 12-p cap 25 01:59 params.h
-rw-rw-r-- 1 dalai dalai 1501 12-p cap 25 03:00 server.c
-rwxrwxr-x 1 dalai dalai 17240 12-p cap 25 03:00 server.out
-rw-rw-r-- 1 dalai dalai 45948 12-p cap 23 15:34 Отчет_лабораторная_6.odt
dalai@dalai-Inspiron-5567:~/Desktop/BMSTU/labs/lab_06$ ./client.out
Sent message: pid 51794
dalai@dalai-Inspiron-5567:~/Desktop/BMSTU/labs/lab_06$ ./client.out
Sent message: pid 51795
dalai@dalai-Inspiron-5567:~/Desktop/BMSTU/labs/lab_06$ ./client.out
Sent message: pid 51799
dalai@dalai-Inspiron-5567:~/Desktop/BMSTU/labs/lab_06$ ./client.out
Sent message: pid 51800
dalai@dalai-Inspiron-5567:~/Desktop/BMSTU/labs/lab_06$ ./client.out
Sent message: pid 51801
dalai@dalai-Inspiron-5567:~/Desktop/BMSTU/labs/lab_06$ ./client.out
Sent message: pid 51802
dalai@dalai-Inspiron-5567:~/Desktop/BMSTU/labs/lab_06$ ^C
```

Рисунок 2. Результаты работы клиента.

2. Задание 2

- Написать приложение по модели клиент-сервер, осуществляющее взаимодействие параллельных процессов, которые выполняются на разных компьютерах. Для взаимодействия с клиентами сервер должен использовать мультиплексирование. Сервер должен обслуживать запросы параллельно запущенных клиентов. При демонстрации работы программного комплекса необходимо запустить продемонстрировать, несколько клиентов сервер обрабатывает что (не меньше обращения 5) и каждого запущенного клиента.

В листингах 5 – 7 представлен код задания. Makefile аналогичен makefile в задании 1.

Листинг 5. Params.h.

```
1 #define MSG_SIZE 256
2 #define SOCKET_PORT 25565
```

Листинг 6. Server.c.

```
1 #include <stdio.h>
2 #include <sys/socket.h>
3 #include <stdbool.h>
4 #include <arpa/inet.h>
5 #include <unistd.h>
6 #include <errno.h>
7 #include "params.h"
8
9 #define MAX_CLIENTS 5
10 #define SOCKET_ERR -1
11 #define BIND_ERR -2
12 #define LISTEN_ERR -3
13 #define ACCEPT_ERR -4
14 #define SELECT_ERR -5
15
16 int clients[MAX_CLIENTS] = {0};
17
18 int set_connection(const int fd) {
19     struct sockaddr_in client_addr;
20     int client_addr_size = sizeof(client_addr);
21     int confd = accept(fd, (struct sockaddr*)&client_addr,
22     (socklen_t *)&client_addr_size);
23     if (confd < 0) {
24         printf("Accept error\n");
25         return ACCEPT_ERR;
26     }
27
28     printf("New connection: FD = %d, IP = %s:%d\n", confd, inet_ntoa(client_addr.sin_addr),
29     ntohs(client_addr.sin_port));
30
31     bool flag = true;
32     for (int i = 0; flag && i < MAX_CLIENTS; i++) {
33         if (clients[i] == 0) {
34             clients[i] = confd;
35             printf("Connected client %d\n", i);
36             flag = false;
37         }
38     }
39     return 0;
40 }
```

```

41
42
43 void get_client_info(const int fd, const int client_id) {
44     struct sockaddr_in client_addr;
45     int client_addr_size = sizeof(client_addr);
46
47     char msg[MSG_SIZE];
48     int sz = recv(fd, msg, MSG_SIZE, 0);
49     if (sz == 0) {
50         getpeername(fd, (struct sockaddr *)&client_addr, (socklen_t *)&client_addr_size);
51         printf("Disconnected: %d, IP = %s:%d\n", client_id,
52               inet_ntoa(client_addr.sin_addr), ntohs(client_addr.sin_port));
53         close(fd);
54         clients[client_id] = 0;
55     }
56     else {
57         msg[sz] = 0;
58         printf("Client %d sent: %s\n", client_id, msg);
59     }
60 }
61
62 int main() {
63     int sockfd = socket(AF_INET, SOCK_STREAM, 0);
64     if (sockfd < 0) {
65         printf("Socket error\n");
66         return SOCKET_ERR;
67     }
68     struct sockaddr_in client_addr = {
69         .sin_family = AF_INET,
70         .sin_port = htons(SOCKET_PORT),
71         .sin_addr.s_addr = INADDR_ANY
72     };
73     if (bind(sockfd, (struct sockaddr*) &client_addr, sizeof(client_addr)) < 0) {
74         printf("Bind error\n");
75         return BIND_ERR;
76     }
77
78     printf("Server %d started successfully, port: %d\n", getpid(), SOCKET_PORT);
79
80     if (listen(sockfd, MAX_CLIENTS) < 0) {
81         printf("Listen error\n");
82         return LISTEN_ERR;
83     }
84
85     while(1) {
86         fd_set readfds;
87         FD_ZERO(&readfds);
88         FD_SET(sockfd, &readfds);
89
90         int max_fd = sockfd;
91         for (int i = 0; i < MAX_CLIENTS; i++) {
92             int fd = clients[i];
93             if (fd > 0)
94                 FD_SET(fd, &readfds);
95             if (fd > max_fd)
96                 max_fd = fd;
97         }
98
99         int active = select(max_fd + 1, &readfds, NULL, NULL, NULL);
100

```



```

101     if ((active < 0) && (errno != EINTR)) {
102         printf("Select error\n");
103         return SELECT_ERR;
104     }
105     if (FD_ISSET(sockfd, &readfds))
106         set_connection(sockfd);
107
108     for (int i = 0; i < MAX_CLIENTS; i++) {
109         int fd = clients[i];
110         if ((fd > 0) && FD_ISSET(fd, &readfds))
111             get_client_info(fd, i);
112     }
113 }
114
115 return 0;
116 }

```

Листинг 7. Client.c.

```

1  #include <sys/socket.h>
2  #include <stdio.h>
3  #include <unistd.h>
4  #include <string.h>
5  #include <stdlib.h>
6  #include <time.h>
7  #include <arpa/inet.h>
8  #include "params.h"
9
10 #define MSG_COUNT 3
11 #define MAX_SLEEP_TIME 3
12 #define SOCKET_ERR -1
13 #define CONNECT_ERR -2
14 #define SEND_ERR -3
15
16 int main() {
17     srand(time(NULL));
18
19     int sockfd = socket(AF_INET, SOCK_STREAM, 0);
20     if (sockfd < 0) {
21         printf("Socket error\n");
22         return SOCKET_ERR;
23     }
24     struct sockaddr_in server_addr = {
25         .sin_family = AF_INET,
26         .sin_port = htons(SOCKET_PORT),
27         .sin_addr.s_addr = INADDR_ANY
28     };
29
30     if (connect(sockfd, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0) {
31         printf("Socket error\n");
32         return CONNECT_ERR;
33     }
34     char msg[MSG_SIZE];
35     for (int i = 0; i < MSG_COUNT; i++) {
36         sprintf(msg, "pid %d msg %d", getpid(), i);
37         if (send(sockfd, msg, strlen(msg), 0) < 0) {
38             printf("Send error\n");

```



```

39         return SEND_ERR;
40     }
41
42     printf("Sent message: %s\n", msg);
43     sleep(1 + rand() % MAX_SLEEP_TIME);
44 }
45 printf("Client end\n");
46 return 0;
47 }

```

На рисунке 3 представлена информация о сервере с помощью команды `lsdf`.

```

dalai@dalai-Inspiron-5567: ~/Desktop/BMSTU/os_labs/lab_06/part_2$ lsdf -i :25565
COMMAND  PID  USER  FD  TYPE  DEVICE  SIZE/OFF  NODE  NAME
server.ou 61333 dalai   3u  IPv4  445362      0t0   TCP *:25565 (LISTEN)
dalai@dalai-Inspiron-5567: ~/Desktop/BMSTU/os_labs/lab_06/part_2$ lsdf -p 61333
COMMAND  PID  USER  FD  TYPE  DEVICE  SIZE/OFF  NODE  NAME
server.ou 61333 dalai   cwd   DIR    8,7      4096   11802045 /home/dalai/Desktop/BMSTU/os_labs/lab_06/part_2
server.ou 61333 dalai   rtd   DIR    8,7      4096     2 /
server.ou 61333 dalai   txt   REG    8,7     17488   11802330 /home/dalai/Desktop/BMSTU/os_labs/lab_06/part_2/server.out
server.ou 61333 dalai   mem   REG    8,7    2029224   2104453 /usr/lib/x86_64-linux-gnu/libc-2.31.so
server.ou 61333 dalai   mem   REG    8,7     191472   2104237 /usr/lib/x86_64-linux-gnu/ld-2.31.so
server.ou 61333 dalai    0u   CHR   136,4      0t0     7 /dev/pts/4
server.ou 61333 dalai    1u   CHR   136,4      0t0     7 /dev/pts/4
server.ou 61333 dalai    2u   CHR   136,4      0t0     7 /dev/pts/4
server.ou 61333 dalai    3u  IPv4  445362      0t0   TCP *:25565 (LISTEN)
dalai@dalai-Inspiron-5567: ~/Desktop/BMSTU/os_labs/lab_06/part_2$

```

Рисунок 3. Информация о сервере.

На рисунке 4 представлена демонстрация работы сервера.

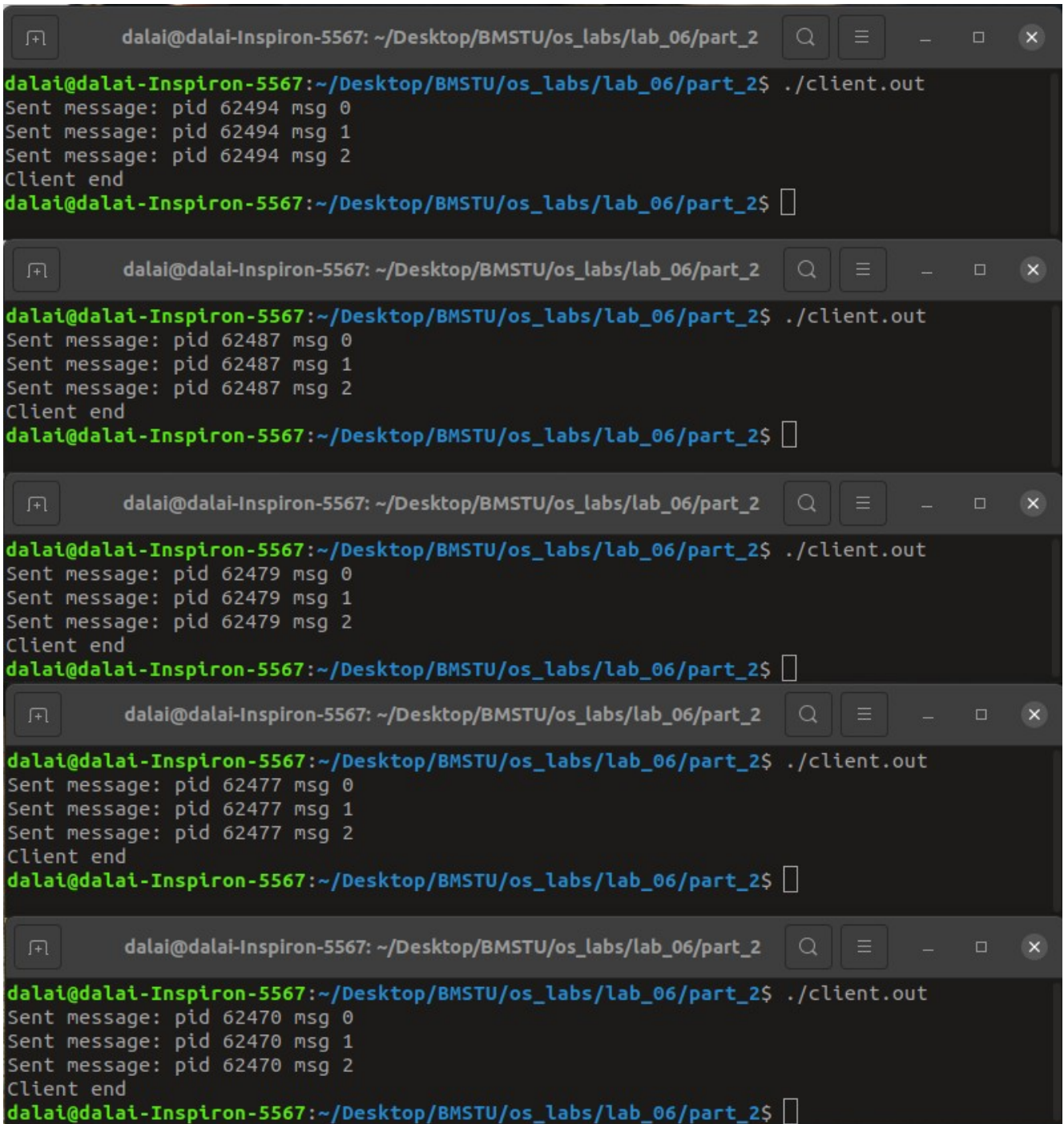
```

dalai@dalai-Inspiron-5567: ~/Desktop/BMSTU/os_labs/lab_06/part_2$ ./server.out
Server 61333 started successfully, port: 25565
New connection: FD = 4, IP = 127.0.0.1:57948
Connected client 0
Client 0 sent: pid 62470 msg 0
Client 0 sent: pid 62470 msg 1
Client 0 sent: pid 62470 msg 2
Disconnected: 0, IP = 127.0.0.1:57948
New connection: FD = 4, IP = 127.0.0.1:57950
Connected client 0
Client 0 sent: pid 62477 msg 0
Client 0 sent: pid 62477 msg 1
Client 0 sent: pid 62477 msg 2
Disconnected: 0, IP = 127.0.0.1:57950
New connection: FD = 4, IP = 127.0.0.1:57952
Connected client 0
Client 0 sent: pid 62479 msg 0
Client 0 sent: pid 62479 msg 1
Client 0 sent: pid 62479 msg 2
Disconnected: 0, IP = 127.0.0.1:57952
New connection: FD = 4, IP = 127.0.0.1:57954
Connected client 0
Client 0 sent: pid 62487 msg 0
Client 0 sent: pid 62487 msg 1
Client 0 sent: pid 62487 msg 2
Disconnected: 0, IP = 127.0.0.1:57954
New connection: FD = 4, IP = 127.0.0.1:57956
Connected client 0
Client 0 sent: pid 62494 msg 0
Client 0 sent: pid 62494 msg 1
Client 0 sent: pid 62494 msg 2
Disconnected: 0, IP = 127.0.0.1:57956

```

Рисунок 4. Демонстрация работы сервера.

На рисунке 5 представлена демонстрация работы клиентов.



The image displays five terminal windows stacked vertically, each showing the output of a client program. The windows have a dark background with light green text for the prompt and white text for the output. The prompt in each window is `dalai@dalai-Inspiron-5567: ~/Desktop/BMSTU/os_labs/lab_06/part_2$`. The output of each window is as follows:

```
dalai@dalai-Inspiron-5567:~/Desktop/BMSTU/os_labs/lab_06/part_2$ ./client.out
Sent message: pid 62494 msg 0
Sent message: pid 62494 msg 1
Sent message: pid 62494 msg 2
Client end
dalai@dalai-Inspiron-5567:~/Desktop/BMSTU/os_labs/lab_06/part_2$
```

```
dalai@dalai-Inspiron-5567:~/Desktop/BMSTU/os_labs/lab_06/part_2$ ./client.out
Sent message: pid 62487 msg 0
Sent message: pid 62487 msg 1
Sent message: pid 62487 msg 2
Client end
dalai@dalai-Inspiron-5567:~/Desktop/BMSTU/os_labs/lab_06/part_2$
```

```
dalai@dalai-Inspiron-5567:~/Desktop/BMSTU/os_labs/lab_06/part_2$ ./client.out
Sent message: pid 62479 msg 0
Sent message: pid 62479 msg 1
Sent message: pid 62479 msg 2
Client end
dalai@dalai-Inspiron-5567:~/Desktop/BMSTU/os_labs/lab_06/part_2$
```

```
dalai@dalai-Inspiron-5567:~/Desktop/BMSTU/os_labs/lab_06/part_2$ ./client.out
Sent message: pid 62477 msg 0
Sent message: pid 62477 msg 1
Sent message: pid 62477 msg 2
Client end
dalai@dalai-Inspiron-5567:~/Desktop/BMSTU/os_labs/lab_06/part_2$
```

```
dalai@dalai-Inspiron-5567:~/Desktop/BMSTU/os_labs/lab_06/part_2$ ./client.out
Sent message: pid 62470 msg 0
Sent message: pid 62470 msg 1
Sent message: pid 62470 msg 2
Client end
dalai@dalai-Inspiron-5567:~/Desktop/BMSTU/os_labs/lab_06/part_2$
```

Рисунок 5. Демонстрация работы клиентов.