



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»

ОТЧЁТ

По лабораторной работе №6

По курсу: «Операционные системы»

Тема: «Сокеты»

Студент: Керимов А. Ш.

Группа: ИУ7-64Б

Преподаватель: Рязанова Н. Ю.

Москва

2020

Часть 1

Задание. 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.

Листинг 1: `unix/socket.h`

```
1 #ifndef SOCKET_H_
2 #define SOCKET_H_
3
4 #define SOCKET_NAME "socket.soc"
5 #define BUF_SIZE 256
6
7 /* defined in linux/kernel.h */
8 #ifndef ARRAY_SIZE
9 #define ARRAY_SIZE(x) (sizeof(x) / sizeof((x)[0]))
10 #endif
11
12 #endif // SOCKET_H_
```

Листинг 2: `unix/server.c`

```
1 #include <signal.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <string.h>
5 #include <sys/socket.h>
6 #include <time.h>
7 #include <unistd.h>
8 #include "socket.h"
9
10 static int sockfd;
11
12 void cleanup(void)
13 {
14     close(sockfd);
15     unlink(SOCKET_NAME);
16 }
17
18 void sighandler(__attribute__((unused)) int signum)
19 {
20     cleanup();
21     exit(EXIT_SUCCESS);
22 }
23
24 void perror_exit(const char *str)
25 {
26     cleanup();
27     perror(str);
28     exit(EXIT_FAILURE);
29 }
30
31 int main(void)
32 {
33     if ((sockfd = socket(AF_UNIX, SOCK_DGRAM, 0)) == -1) {
34         perror("socket");
35         return EXIT_FAILURE;
36     }
37
38     struct sockaddr addr;
39     addr.sa_family = AF_UNIX;
40     strncpy(addr.sa_data, SOCKET_NAME, ARRAY_SIZE(addr.sa_data));
41     if (bind(sockfd, &addr, sizeof addr) == -1) {
42         perror_exit("bind");
43     }
44
45     if (signal(SIGINT, sighandler) == SIG_ERR) {
46         perror_exit("signal");
47     }
48 }
```

```

49     while (1) {
50         char msg[BUF_SIZE];
51         const ssize_t bytes = recv(sockfd, msg, sizeof msg, 0);
52         if (bytes == -1) {
53             perror_exit("recv");
54         }
55         msg[bytes] = '\0';
56
57         const time_t timer = time(NULL);
58         printf("[%19s] receive message: %s\n", ctime(&timer), msg);
59     }
60 }

```

Листинг 3: unix/client.c

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <sys/socket.h>
5  #include <unistd.h>
6  #include <time.h>
7  #include "socket.h"
8
9  int main(void)
10 {
11     int rc = EXIT_SUCCESS;
12
13     const int sockfd = socket(AF_UNIX, SOCK_DGRAM, 0);
14     if (sockfd == -1) {
15         perror("socket(AF_UNIX, SOCK_DGRAM, 0)");
16         return EXIT_FAILURE;
17     }
18
19     struct sockaddr addr;
20     addr.sa_family = AF_UNIX;
21     strncpy(addr.sa_data, SOCKET_NAME, ARRAY_SIZE(addr.sa_data));
22
23     char msg[BUF_SIZE];
24     snprintf(msg, ARRAY_SIZE(msg), "pid %d", getpid());
25     if (sendto(sockfd, msg, strlen(msg), 0, &addr, sizeof addr) == -1) {
26         perror("sendto");
27         rc = EXIT_FAILURE;
28     } else {
29         const time_t timer = time(NULL);
30         printf("[%19s] send message: %s\n", ctime(&timer), msg);
31     }
32
33     close(sockfd);
34     return rc;
35 }

```

```

user@lenovo: ~/bmstu/OS/sem06/lab06/unix
user@lenovo:~/bmstu/OS/sem06/lab06/unix$ ./server
[Sun Apr 26 21:27:17] receive message: pid 10099
[Sun Apr 26 21:27:21] receive message: pid 10102
[Sun Apr 26 21:27:25] receive message: pid 10103
[Sun Apr 26 21:27:28] receive message: pid 10104
[Sun Apr 26 21:27:31] receive message: pid 10105
user@lenovo:~/bmstu/OS/sem06/lab06/unix$

user@lenovo:~/bmstu/OS/sem06/lab06/unix$ ./client
[Sun Apr 26 21:27:17] send message: pid 10099
user@lenovo:~/bmstu/OS/sem06/lab06/unix$

user@lenovo:~/bmstu/OS/sem06/lab06/unix$ ./client
[Sun Apr 26 21:27:25] send message: pid 10103
user@lenovo:~/bmstu/OS/sem06/lab06/unix$

user@lenovo:~/bmstu/OS/sem06/lab06/unix$ ./client
[Sun Apr 26 21:27:28] send message: pid 10104
user@lenovo:~/bmstu/OS/sem06/lab06/unix$

user@lenovo:~/bmstu/OS/sem06/lab06/unix$ ./client
[Sun Apr 26 21:27:31] send message: pid 10105
user@lenovo:~/bmstu/OS/sem06/lab06/unix$

```

Рис. 1: Демонстрация работы программы

Часть 2

Задание. 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 несколько клиентов (не меньше 5) и продемонстрировать, что сервер обрабатывает обращения каждого запущенного клиента.

Листинг 4: net/socket.h

```
1 #ifndef SOCKET_H_
2 #define SOCKET_H_
3
4 #define SOCEKT_PORT 8888
5 #define BUF_SIZE 256
6
7 /* defined in linux/kernel.h */
8 #ifndef ARRAY_SIZE
9 #define ARRAY_SIZE(x) (sizeof(x) / sizeof((x)[0]))
10 #endif
11
12 #endif // SOCKET_H_
```

Листинг 5: net/server.c

```
1 #include <arpa/inet.h>
2 #include <netinet/in.h>
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <string.h>
6 #include <sys/socket.h>
7 #include <time.h>
8 #include <unistd.h>
9 #include "socket.h"
10
11 #define MAX_CLIENTS_COUNT 10
12
13 static int master_sd;
14 static int clients[MAX_CLIENTS_COUNT];
15
16 int perror_exit(const char *str)
17 {
18     close(master_sd);
19     perror(str);
20     exit(EXIT_FAILURE);
21 }
22
23 void handle_connection(void)
24 {
25     const int sd = accept(master_sd, NULL, NULL);
26     if (sd == -1) {
27         perror_exit("accept");
28     }
29
30     const time_t timer = time(NULL);
31     for (int i = 0; i < MAX_CLIENTS_COUNT; ++i) {
32         if (!clients[i]) {
33             clients[i] = sd;
34             printf("[%s] new connection: client %d (sd = %d)\n", ctime(&timer), i, sd);
35             return;
36         }
37     }
38
39     close(master_sd);
40     fprintf(stderr, "[%s] reached MAX_CLIENTS_COUNT\n", ctime(&timer));
41     exit(EXIT_FAILURE);
42 }
43
44 void handle_client(int i)
45 {
46     char msg[BUF_SIZE];
47     const ssize_t bytes = recv(clients[i], &msg, ARRAY_SIZE(msg), 0);
```

```

48     const time_t timer = time(NULL);
49     if (!bytes) {
50         printf("[%19s] client %d (sd = %d) disconnected\n", ctime(&timer), i, clients[i]);
51         close(clients[i]);
52         clients[i] = 0;
53         return;
54     }
55
56     msg[bytes] = '\0';
57     printf("[%19s] receive message from client %d (sd = %d): %s\n", ctime(&timer), i, clients[i],
58         msg);
59 }
60 int main(void)
61 {
62     if ((master_sd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
63         perror("socket");
64         return EXIT_FAILURE;
65     }
66
67     struct sockaddr_in addr = {
68         .sin_family = AF_INET,
69         .sin_addr.s_addr = INADDR_ANY,
70         .sin_port = htons(SOCKET_PORT)
71     };
72
73     if (bind(master_sd, (struct sockaddr *) &addr, sizeof addr) == -1) {
74         perror_exit("bind");
75     }
76
77     if (listen(master_sd, MAX_CLIENTS_COUNT) == -1) {
78         perror_exit("listen");
79     }
80
81     const time_t timer = time(NULL);
82     printf("[%19s] server is running on %s:%d\n", ctime(&timer), inet_ntoa(addr.sin_addr),
83         ntohs(addr.sin_port));
84
85     while (1) {
86         fd_set readfds;
87         FD_ZERO(&readfds);
88
89         FD_SET(master_sd, &readfds);
90
91         int max_sd = master_sd;
92         for (int i = 0; i < MAX_CLIENTS_COUNT; ++i) {
93             if (clients[i] > 0) {
94                 FD_SET(clients[i], &readfds);
95             }
96             if (clients[i] > max_sd) {
97                 max_sd = clients[i];
98             }
99         }
100
101         if (select(max_sd + 1, &readfds, NULL, NULL, NULL) == -1) {
102             perror_exit("select");
103         }
104
105         if (FD_ISSET(master_sd, &readfds)) {
106             handle_connection();
107         }
108
109         for (int i = 0; i < MAX_CLIENTS_COUNT; ++i) {
110             if (clients[i] && FD_ISSET(clients[i], &readfds)) {
111                 handle_client(i);
112             }
113         }
114     }

```

Листинг 6: net/client.c

```

1 #include <netdb.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <string.h>
5 #include <sys/socket.h>
6 #include <time.h>
7 #include <unistd.h>

```

```

8 #include "socket.h"
9
10 int main(void)
11 {
12     const int master_sd = socket(AF_INET, SOCK_STREAM, 0);
13     if (master_sd == -1) {
14         perror("socket");
15         return EXIT_FAILURE;
16     }
17
18     struct sockaddr_in addr = {
19         .sin_family = AF_INET,
20         .sin_addr.s_addr = INADDR_ANY,
21         .sin_port = htons(SOCKET_PORT)
22     };
23
24     if (connect(master_sd, (struct sockaddr *) &addr, sizeof addr) == -1) {
25         perror("connect");
26         return EXIT_FAILURE;
27     }
28
29     do {
30         char msg[BUF_SIZE];
31         snprintf(msg, ARRAY_SIZE(msg), "pid %d", getpid());
32         if (sendto(master_sd, msg, strlen(msg), 0, (struct sockaddr *) &addr, sizeof addr) ==
33             -1) {
34             perror("sendto");
35             return EXIT_FAILURE;
36         }
37
38         const time_t timer = time(NULL);
39         printf("[%s] sent message: %s\n", ctime(&timer), msg);
40         printf("Press Enter to resend message...\n");
41         getchar();
42     } while (1);
43 }

```

```

user@lenovo:~/bmstu/OS/sen06/lab06/net$ ./server
[Sun Apr 26 21:28:11] server is running on 0.0.0.0:8888
[Sun Apr 26 21:28:15] new connection: client 0 (sd = 4): pid 10140
[Sun Apr 26 21:28:15] receive message from client 0 (sd = 4): pid 10140
[Sun Apr 26 21:28:19] new connection: client 1 (sd = 5): pid 10146
[Sun Apr 26 21:28:19] receive message from client 1 (sd = 5): pid 10146
[Sun Apr 26 21:28:22] new connection: client 2 (sd = 6): pid 10147
[Sun Apr 26 21:28:22] receive message from client 2 (sd = 6): pid 10147
[Sun Apr 26 21:28:27] new connection: client 3 (sd = 7): pid 10148
[Sun Apr 26 21:28:27] receive message from client 3 (sd = 7): pid 10148
[Sun Apr 26 21:28:35] new connection: client 4 (sd = 8): pid 10149
[Sun Apr 26 21:28:35] receive message from client 4 (sd = 8): pid 10149
[Sun Apr 26 21:28:40] receive message from client 0 (sd = 4): pid 10140
[Sun Apr 26 21:28:40] receive message from client 1 (sd = 5): pid 10146
[Sun Apr 26 21:28:41] receive message from client 2 (sd = 6): pid 10147
[Sun Apr 26 21:28:41] receive message from client 3 (sd = 7): pid 10148
[Sun Apr 26 21:28:42] receive message from client 4 (sd = 8): pid 10149
[Sun Apr 26 21:28:46] client 0 (sd = 4) disconnected
[Sun Apr 26 21:28:46] client 1 (sd = 5) disconnected
[Sun Apr 26 21:28:47] client 2 (sd = 6) disconnected
[Sun Apr 26 21:28:47] client 3 (sd = 7) disconnected
[Sun Apr 26 21:28:48] client 4 (sd = 8) disconnected

user@lenovo:~/bmstu/OS/sen06/lab06/net$ ./client
[Sun Apr 26 21:28:15] sent message: pid 10140
Press Enter to resend message...
[Sun Apr 26 21:28:40] sent message: pid 10140
Press Enter to resend message...^C
user@lenovo:~/bmstu/OS/sen06/lab06/net$

user@lenovo:~/bmstu/OS/sen06/lab06/net$ ./client
[Sun Apr 26 21:28:22] sent message: pid 10147
Press Enter to resend message...
[Sun Apr 26 21:28:41] sent message: pid 10147
Press Enter to resend message...^C
user@lenovo:~/bmstu/OS/sen06/lab06/net$

user@lenovo:~/bmstu/OS/sen06/lab06/net$ ./client
[Sun Apr 26 21:28:35] sent message: pid 10149
Press Enter to resend message...
[Sun Apr 26 21:28:42] sent message: pid 10149
Press Enter to resend message...^C
user@lenovo:~/bmstu/OS/sen06/lab06/net$

```

Рис. 2: Демонстрация работы программы