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`

```h
#ifndef SOCKET_H_
#define SOCKET_H_

#define SOCKET_NAME "socket.soc"
#define BUF_SIZE 256

/* defined in linux/kernel.h */
#ifndef ARRAY_SIZE
#define ARRAY_SIZE(x) (sizeof(x) / sizeof((x)[0]))
#endif

#endif  // SOCKET_H_
```

Листинг 2: `unix/server.c`

```c
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <time.h>
#include <unistd.h>
#include "socket.h"

static int sockfd;

void cleanup(void)
{
        close(sockfd);
        unlink(SOCKET_NAME);
}

void sighandler(__attribute__((unused)) int  signum)
{
        cleanup();
        exit(EXIT_SUCCESS);
}

void perror_exit(const char *str)
{
        cleanup();
        perror(str);
        exit(EXIT_FAILURE);
}

int main(void)
{
        if ((sockfd = socket(AF_UNIX, SOCK_DGRAM, 0)) == -1) {
                perror("socket");
                return EXIT_FAILURE;
        }

        struct sockaddr addr;
        addr.sa_family = AF_UNIX;
        strncpy(addr.sa_data, SOCKET_NAME, ARRAY_SIZE(addr.sa_data));
        if (bind(sockfd, &addr, sizeof addr) == -1) {
                perror_exit("bind");
        }

        if (signal(SIGINT, sighandler) == SIG_ERR) {
                perror_exit("signal");
        }

```

```
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  }
```

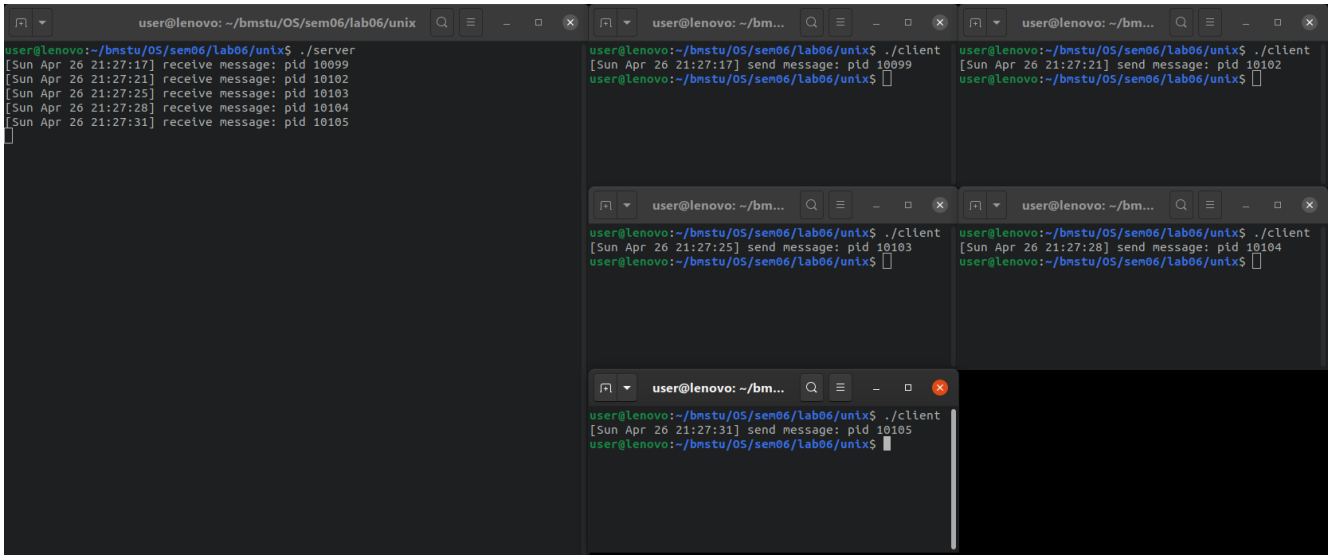

Рис. 1: Демонстрация работы программы

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`

```h
#ifndef SOCKET_H_
#define SOCKET_H_

#define SOCEKT_PORT 8888
#define BUF_SIZE 256

/* defined in linux/kernel.h */
#ifndef ARRAY_SIZE
#define ARRAY_SIZE(x) (sizeof(x) / sizeof((x)[0]))
#endif

#endif  // SOCKET_H_
```

Листинг 5: `net/server.c`

```c
#include <arpa/inet.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <time.h>
#include <unistd.h>
#include "socket.h"

#define MAX_CLIENTS_COUNT 10

static int master_sd;
static int clients[MAX_CLIENTS_COUNT];

int perror_exit(const char *str)
{
        close(master_sd);
        perror(str);
        exit(EXIT_FAILURE);
}

void handle_connection(void)
{
        const int sd = accept(master_sd, NULL, NULL);
        if (sd == -1) {
                perror_exit("accept");
        }

        const time_t timer = time(NULL);
        for (int i = 0; i < MAX_CLIENTS_COUNT; ++i) {
                if (!clients[i]) {
                        clients[i] = sd;
                        printf("[%.19s] new connection: client %d (sd = %d)\n", ctime(&timer), i, sd);
                        return;
                }
        }

        close(master_sd);
        fprintf(stderr, "[%.19s] reached MAX_CLIENTS_COUNT\n", ctime(&timer));
        exit(EXIT_FAILURE);
}

void handle_client(int i)
{
        char msg[BUF_SIZE];
        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],
                msg);
58  }
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(SOCEKT_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),
                ntohs(addr.sin_port));
83
84          while (1) {
85                  fd_set readfds;
86                  FD_ZERO(&readfds);
87
88                  FD_SET(master_sd, &readfds);
89
90                  int max_sd = master_sd;
91                  for (int i = 0; i < MAX_CLIENTS_COUNT; ++i) {
92                          if (clients[i] > 0) {
93                                  FD_SET(clients[i], &readfds);
94                          }
95                          if (clients[i] > max_sd) {
96                                  max_sd = clients[i];
97                          }
98                  }
99
100                 if (select(max_sd + 1, &readfds, NULL, NULL, NULL) == -1) {
101                         perror_exit("select");
102                 }
103
104                 if (FD_ISSET(master_sd, &readfds)) {
105                         handle_connection();
106                 }
107
108                 for (int i = 0; i < MAX_CLIENTS_COUNT; ++i) {
109                         if (clients[i] && FD_ISSET(clients[i], &readfds)) {
110                                 handle_client(i);
111                         }
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>
```

```c
#include "socket.h"

int main(void)
{
        const int master_sd = socket(AF_INET, SOCK_STREAM, 0);
        if (master_sd == -1) {
                perror("socket");
                return EXIT_FAILURE;
        }

        struct sockaddr_in addr = {
                .sin_family = AF_INET,
                .sin_addr.s_addr = INADDR_ANY,
                .sin_port = htons(SOCEKT_PORT)
        };

        if (connect(master_sd, (struct sockaddr *) &addr, sizeof addr) == -1) {
                perror("connect");
                return EXIT_FAILURE;
        }

        do {
                char msg[BUF_SIZE];
                snprintf(msg, ARRAY_SIZE(msg), "pid %d", getpid());
                if (sendto(master_sd, msg, strlen(msg), 0, (struct sockaddr *) &addr, sizeof addr) ==
                    -1) {
                        perror("sendto");
                        return EXIT_FAILURE;
                }

                const time_t timer = time(NULL);
                printf("[%.19s] sent message: %s\n", ctime(&timer), msg);
                printf("Press Enter to resend message...");
                getchar();
        } while (1);
}
```

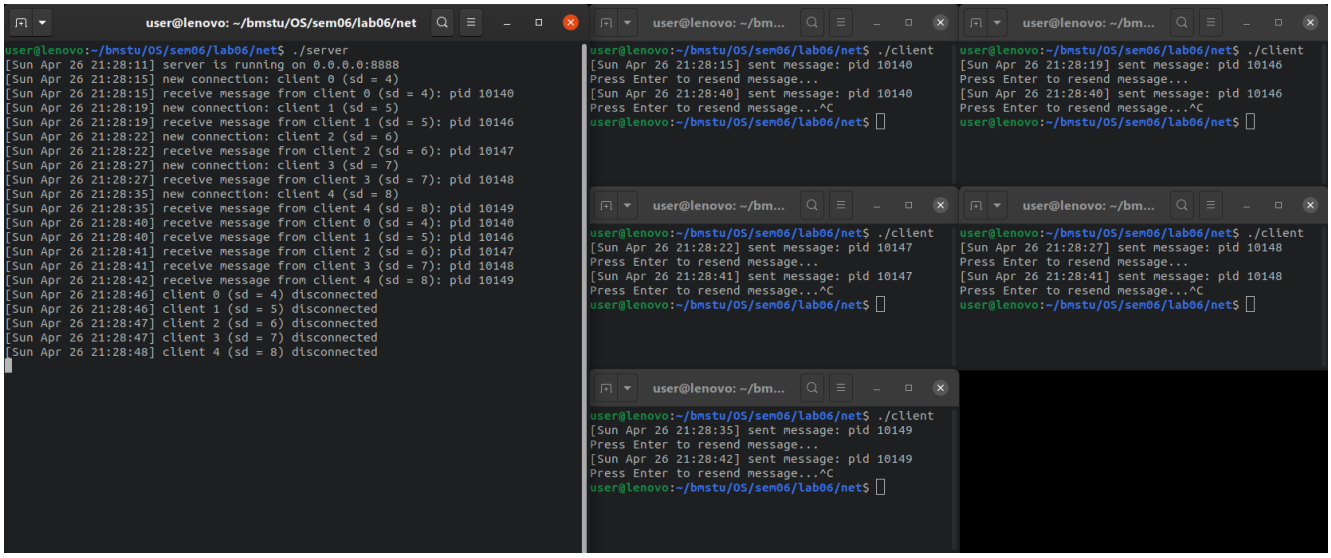

Рис. 2: Демонстрация работы программы