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»

О Т Ч Е Т

по лабораторной работе № ____

Дисциплина: ***Операционные системы***

Студент	***ИУ7И-66Б***	________________	**Нгуен Ф. С.**
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель		________________	**Рязанова Н. Ю.**
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2021

I. Первая часть (AF_UNIX)

Server.c

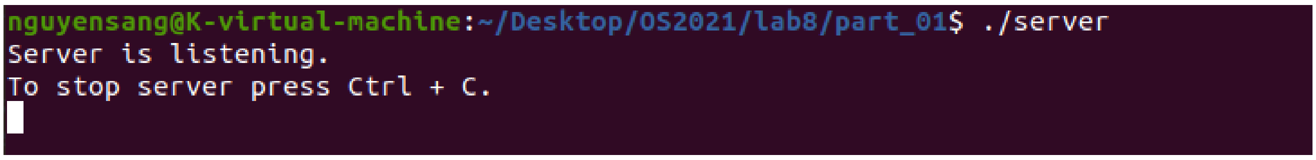
```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/socket.h>

#define SOCKET_NAME "./socket"
#define BUF_SIZE 256
#define OK 0

static int sockfd;

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

void sigint_handler(int signum)
{
    cleanup_socket();
    exit(OK);
}

int main(void)
{
    if ((sockfd = socket(AF_UNIX, SOCK_DGRAM, 0)) < 0)
    {
        perror("Failed to create socket");
        return EXIT_FAILURE;
    }

    struct sockaddr srvr_name;
    srvr_name.sa_family = AF_UNIX;
    strcpy(srvr_name.sa_data, SOCKET_NAME);
    if (bind(sockfd, &srvr_name, strlen(srvr_name.sa_data) +
sizeof(srvr_name.sa_family)) < 0)
    {
        perror("Failed to bind socket");
        return EXIT_FAILURE;
    }

    signal(SIGINT, sigint_handler);
    fprintf(stdout, "Server is listening.\nTo stop server press Ctrl + C.\n");

    char buf[BUF_SIZE];
    for (;;)
    {
        int bytes = recv(sockfd, buf, sizeof(buf), 0);
        if (bytes <= 0)
        {
            perror("Failed to recv");
            cleanup_socket();
            return EXIT_FAILURE;
        }

        buf[bytes] = '\0';
        fprintf(stdout, "Server read: [%s]\n", buf);
    }
    fprintf(stdout, "Server stopped listening\n");
    cleanup_socket();
    fprintf(stdout, "Socket closed\n");

    return OK;
}
```

Client.c

```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>

#define SOCKET_NAME "./socket"
#define BUF_SIZE 256
#define OK 0

int main(void)
{
    int sockfd = socket(AF_UNIX, SOCK_DGRAM, 0);
    if (sockfd < 0)
    {
        perror("Failed to create socket");
        return EXIT_FAILURE;
    }

    struct sockaddr srvr_name;
    srvr_name.sa_family = AF_UNIX;
    strcpy(srvr_name.sa_data, SOCKET_NAME);

    char buf[BUF_SIZE];
    snprintf(buf, BUF_SIZE, "This Message From %d", getpid());
    if (sendto(sockfd, buf, strlen(buf), 0, &srvr_name, strlen(srvr_name.sa_data)
+ sizeof(srvr_name.sa_family)) < 0)
    {
        perror("Failed to send message");
        close(sockfd);
        return EXIT_FAILURE;
    }

    printf("Client sent: [%s]\n", buf);
    return OK;
}
```

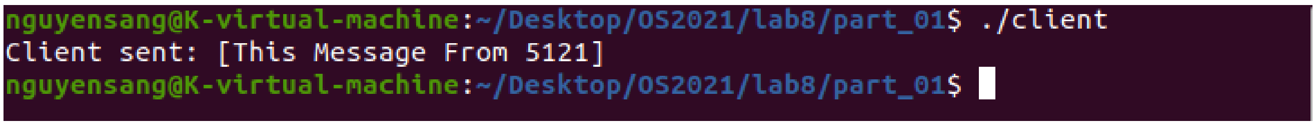
Результат

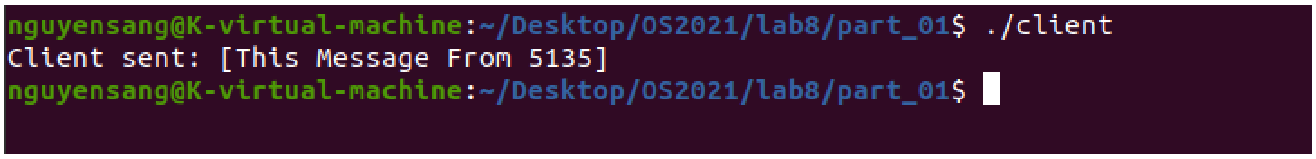
$ ./server



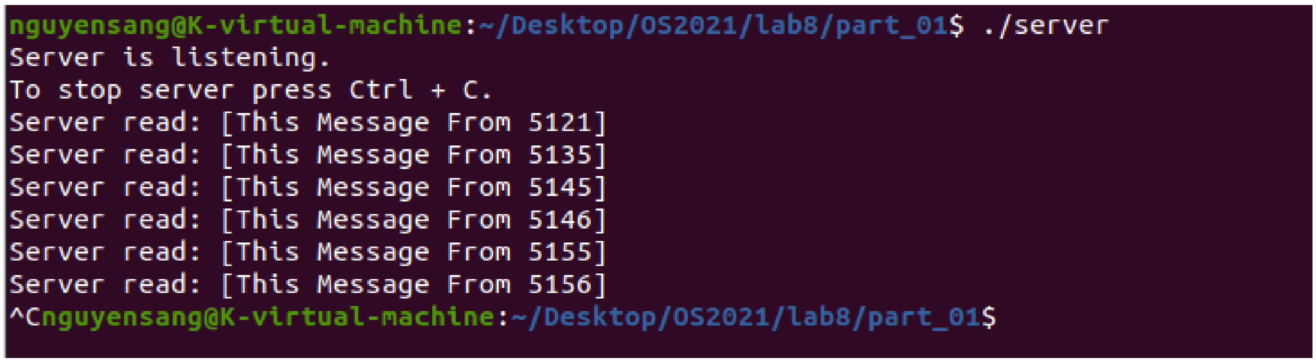
```
nguyensang@K-virtual-machine:~/Desktop/OS2021/lab8/part_01$ ./server
Server is listening.
To stop server press Ctrl + C.
```

$ ./Client


```
nguyensang@K-virtual-machine:~/Desktop/OS2021/lab8/part_01$ ./client
Client sent: [This Message From 5121]
nguyensang@K-virtual-machine:~/Desktop/OS2021/lab8/part_01$ 
```


```
nguyensang@K-virtual-machine:~/Desktop/OS2021/lab8/part_01$ ./client
Client sent: [This Message From 5135]
nguyensang@K-virtual-machine:~/Desktop/OS2021/lab8/part_01$ 
```

…..


```
nguyensang@K-virtual-machine:~/Desktop/OS2021/lab8/part_01$ ./server
Server is listening.
To stop server press Ctrl + C.
Server read: [This Message From 5121]
Server read: [This Message From 5135]
Server read: [This Message From 5145]
Server read: [This Message From 5146]
Server read: [This Message From 5155]
Server read: [This Message From 5156]
^Cnguyensang@K-virtual-machine:~/Desktop/OS2021/lab8/part_01$
```

Ii. Вторая часть (AF_INET)

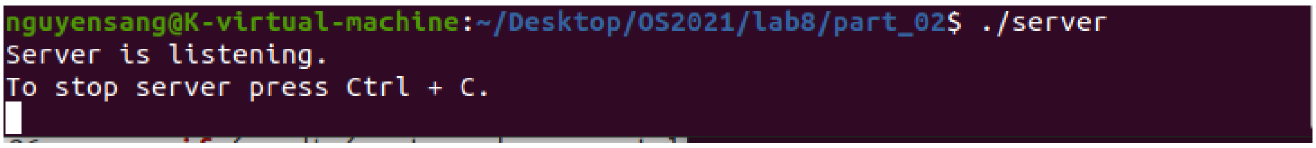
server.c

```
#include "socket.h"

#define MAX_CLIENTS_COUNT 10

static int master_sd;
static int clients[MAX_CLIENTS_COUNT];

int cleanup()
{
    close(master_sd);
    exit(EXIT_FAILURE);
}

void sigint_handler(int signum)
{
    cleanup();
    exit(OK);
}

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

    for (int i = 0; i < MAX_CLIENTS_COUNT; ++i)
    {
        if (!clients[i])
        {
            clients[i] = sd;
            fprintf(stdout, "New connection.\n");
            return;
        }
    }

    fprintf(stderr, "Reached MAX_CLIENTS_COUNT (%d)\n",
MAX_CLIENTS_COUNT);
    cleanup();
}

void handle_client(int i)
{
    char msg[BUF_SIZE];
    const ssize_t bytes = recv(clients[i], &msg, BUF_SIZE, 0);

    if (!bytes)
    {
        close(clients[i]);
        clients[i] = 0;
        return;
    }

    msg[bytes] = '\0';
    fprintf(stdout, "Get message from client: %s\n", msg);
}

int main(void)
{
    if ((master_sd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        perror("Failed to create socket");
        return EXIT_FAILURE;
    }

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

    if (bind(master_sd, (struct sockaddr *) &addr, sizeof addr) < 0)
    {
        cleanup();
    }

    if (listen(master_sd, MAX_CLIENTS_COUNT) < 0)
    {
        cleanup();
    }

    signal(SIGINT, sigint_handler);
    fprintf(stdout, "Server is listening.\nTo stop server press Ctrl +
C.\n");

    while (1)
    {
        fd_set readfds;
        FD_ZERO(&readfds);
        FD_SET(master_sd, &readfds);
        int max_sd = master_sd;

        for (int i = 0; i < MAX_CLIENTS_COUNT; ++i)
        {
            if (clients[i] > 0)
            {
                FD_SET(clients[i], &readfds);
            }

            if (clients[i] > max_sd)
            {
                max_sd = clients[i];
            }
        }

        if (pselect(max_sd + 1, &readfds, NULL, NULL, NULL, NULL) < 0)
        {
            cleanup();
            perror("Failed to select");
        }

        if (FD_ISSET(master_sd, &readfds))
        {
            handle_connection();
        }

        for (int i = 0; i < MAX_CLIENTS_COUNT; ++i)
        {
            if (clients[i] && FD_ISSET(clients[i], &readfds))
            {
                handle_client(i);
            }
        }
    }
}
```

Client.c

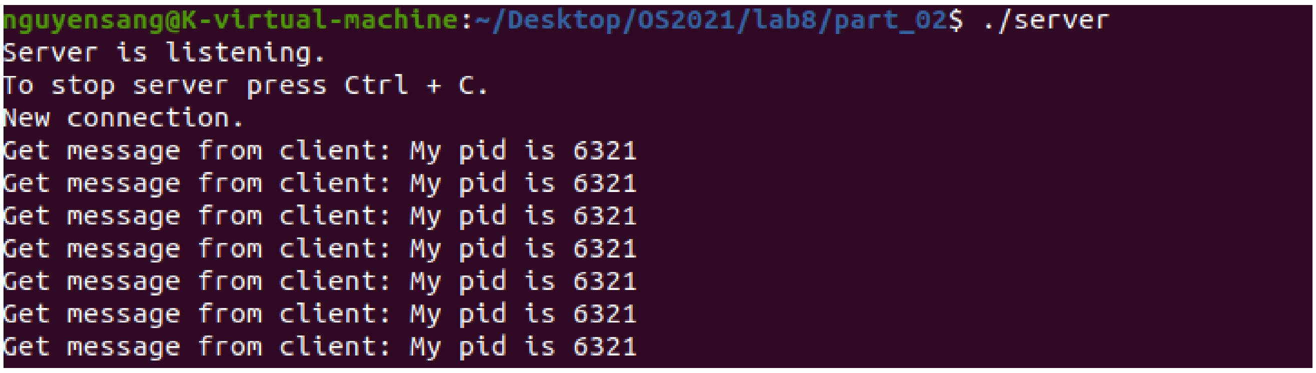
```
include "socket.h"

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

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

    if (connect(master_sd, (struct sockaddr *) &addr, sizeof addr) < 0) {
        perror("Failed to connect");
        return EXIT_FAILURE;
    }

    while (1)
    {
        char msg[BUF_SIZE];
        snprintf(msg, BUF_SIZE, "My pid is %d", getpid());
        if (sendto(master_sd, msg, strlen(msg), 0, (struct sockaddr *)
&addr, sizeof addr) < 0)
        {
            perror("Failed to sendto");
            return EXIT_FAILURE;
        }

        sleep(1);
    }
}
```

$ ./server



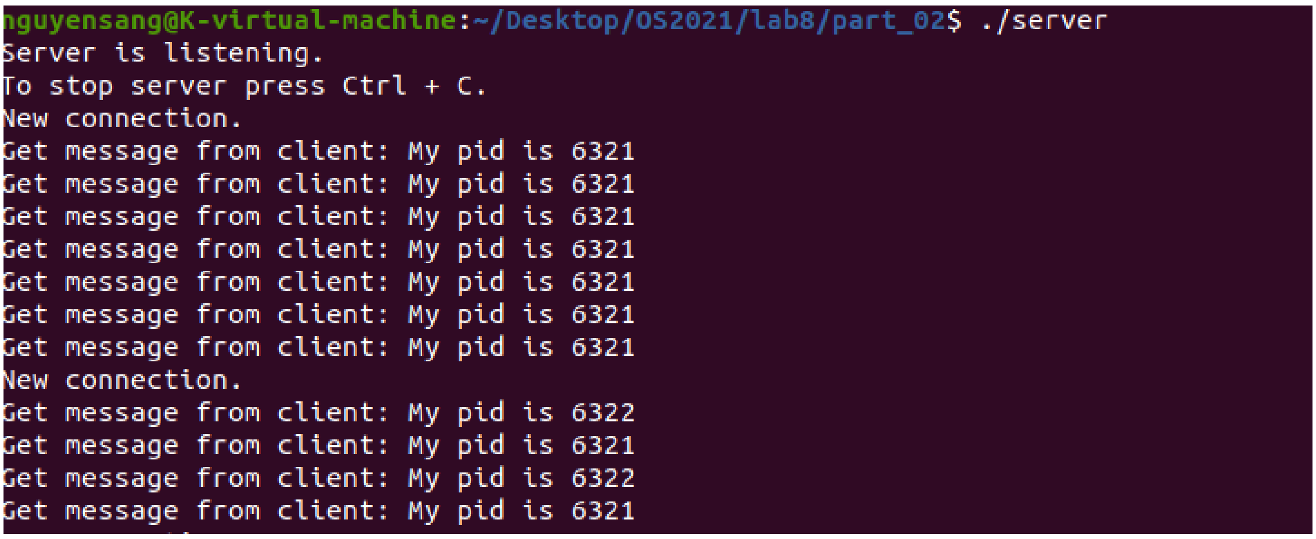
```
nguyensang@K-virtual-machine:~/Desktop/OS2021/lab8/part_02$ ./server
Server is listening.
To stop server press Ctrl + C.
```

$ ./client



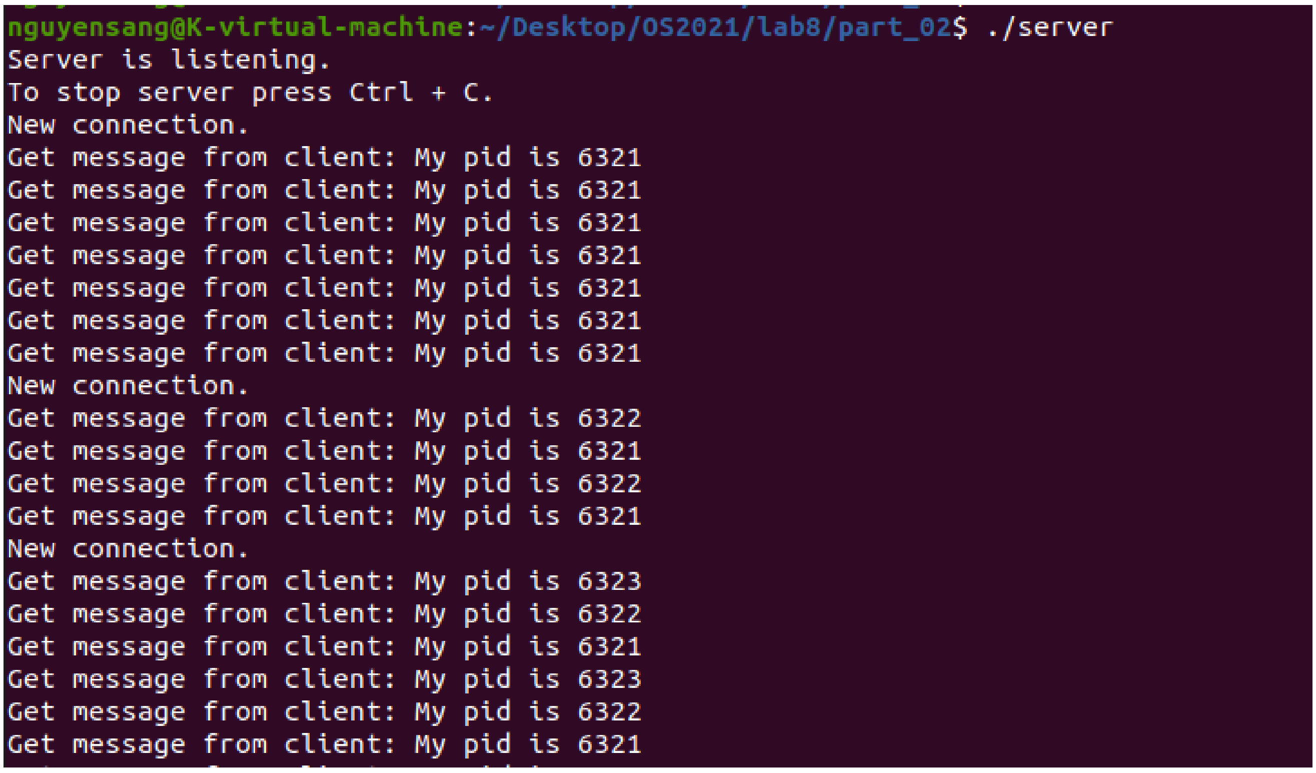
```
nguyensang@K-virtual-machine:~/Desktop/OS2021/lab8/part_02$ ./server
Server is listening.
To stop server press Ctrl + C.
New connection.
Get message from client: My pid is 6321
Get message from client: My pid is 6321
Get message from client: My pid is 6321
Get message from client: My pid is 6321
Get message from client: My pid is 6321
Get message from client: My pid is 6321
Get message from client: My pid is 6321
```

$ ./client



```
nguyensang@K-virtual-machine:~/Desktop/OS2021/lab8/part_02$ ./server
Server is listening.
To stop server press Ctrl + C.
New connection.
Get message from client: My pid is 6321
Get message from client: My pid is 6321
Get message from client: My pid is 6321
Get message from client: My pid is 6321
Get message from client: My pid is 6321
Get message from client: My pid is 6321
Get message from client: My pid is 6321
New connection.
Get message from client: My pid is 6322
Get message from client: My pid is 6321
Get message from client: My pid is 6322
Get message from client: My pid is 6321
```

$ ./client



```
nguyensang@K-virtual-machine:~/Desktop/OS2021/lab8/part_02$ ./server
Server is listening.
To stop server press Ctrl + C.
New connection.
Get message from client: My pid is 6321
Get message from client: My pid is 6321
Get message from client: My pid is 6321
Get message from client: My pid is 6321
Get message from client: My pid is 6321
Get message from client: My pid is 6321
Get message from client: My pid is 6321
New connection.
Get message from client: My pid is 6322
Get message from client: My pid is 6321
Get message from client: My pid is 6322
Get message from client: My pid is 6321
New connection.
Get message from client: My pid is 6323
Get message from client: My pid is 6322
Get message from client: My pid is 6321
Get message from client: My pid is 6323
Get message from client: My pid is 6322
Get message from client: My pid is 6321
```