

Министерство высшего образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)
Кафедра безопасности информационных систем (БИС)

СОКЕТЫ

Отчет по лабораторной работе №7
по дисциплине «Системное программирование»

Студентка гр.737-1

_____ Агеева В.С.

____.____.2021г

Принял

Руководитель

доцент кафедры БИС

_____ Романов А.С.

____.____.2021г

Томск 2021

1 Введение

Цель работы: Познакомиться с основными аспектами работы с сокетами. Познакомиться с соответствующими функциями WinAPI и POSIXAPI.

Задание:

1. Изучить краткие теоретические сведения и лекционный материал по теме практического задания.
2. Реализовать приведенные примеры программ.
3. Самостоятельно изучить средства программирования сокетов в ОС Windows и отразить в отчете.
4. Реализовать примеры клиентских программ под ОС Windows для обмена сообщениями с серверами TCP и UDP для Unix/Linux.
5. Написать отчет и защитить у преподавателя.

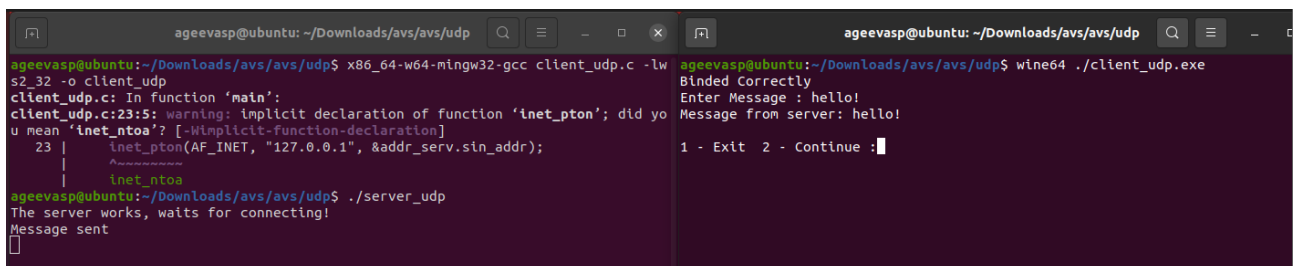
2 Ход работы

В качестве первого знакомства с сокетами были реализованы примеры клиентов и серверов в доменах internet и unix.

В ходе работы был написан сервер в internet домене и клиент, которые для связи используют потоковые сокеты. Сервер связывает два клиента и обеспечивает двустороннюю связь между ними: на сервере выполняются два потока, первый получает сообщения от первого клиента и отправляет их второму, второй поток – получает сообщение от второго клиента и отправляет их первому. Для запуска сервера в качестве аргументов нужно передать два порта – для связи с двумя клиентами соответственно.

Каждый из клиентов поддерживает отправку и получение сообщений: для отправки сообщения нужно выбрать соответствующий пункт, при этом на клиенте параллельно запущен второй поток, который слушает входящие сообщения и выводит их в консоль.

На рисунках 2.1 – 2.2 показаны результаты работы программ.



```
ageevasp@ubuntu: ~/Downloads/avs/avs/udp
ageevasp@ubuntu:~/Downloads/avs/avs/udp$ x86_64-w64-mingw32-gcc client_udp.c -ls2_32 -o client_udp
client_udp.c: In function 'main':
client_udp.c:23:5: warning: implicit declaration of function 'inet_pton'; did you mean 'inet_ntoa'? [-Wimplicit-function-declaration]
   23 |     inet_pton(AF_INET, "127.0.0.1", &addr_serv.sin_addr);
      |     ~~~~~
   24 |     inet_ntoa
ageevasp@ubuntu:~/Downloads/avs/avs/udp$ ./server_udp
The server works, waits for connecting!
Message sent
[ ]

ageevasp@ubuntu:~/Downloads/avs/avs/udp$ wine64 ./client_udp.exe
Binded Correctly
Enter Message : hello!
Message from server: hello!
1 - Exit 2 - Continue : [ ]
```

Рисунок 2.1 – Запуск сервера и отправка сообщения от клиента на UDP

The image shows two terminal windows side-by-side. The left window shows the Docker build process for a TCP server. The right window shows the compilation and execution of a client program.

```

ageevasp@ubuntu: ~/Downloads/avs/avs
Sending build context to Docker daemon 672.3kB
Step 1/5 : FROM debian
--> dc2eddc15825
Step 2/5 : WORKDIR /lab
--> Using cache
--> 685eacbb2d25
Step 3/5 : COPY tcp/server_tcp.c udp/server_udp.c /lab/
--> Using cache
--> 27b74fa2732a
Step 4/5 : RUN apt update;
--> Using cache
--> ee76ff33fed4
Step 5/5 : CMD gcc server_tcp.c -lpthread -o server_tcp;
--> Using cache
--> d245bc616494
Successfully built d245bc616494
Successfully tagged server:latest
ageevasp@ubuntu:~/Downloads/avs/avs$ docker run -it -p 1234:1234 server
root@f7ec0908f73e:/lab# ./server_tcp
The server works, waits for connecting!
The connection is created!

ageevasp@ubuntu: ~/Downloads/avs/avs/tcp
ageevasp@ubuntu:~/Downloads/avs/avs/tcp$ x86_64-w64-mingw32-gcc client_tcp.c -lw
s2_32 -o client_tcp
client_tcp.c: In function 'main':
client_tcp.c:21:5: warning: implicit declaration of function 'inet_pton'; did you
u mean 'inet_ntoa'? [-Wimplicit-function-declaration]
    21 |     inet_pton(AF_INET, "127.0.0.1", &addr_serv.sin_addr);
        |     ~~~~~
        |     inet_ntoa
ageevasp@ubuntu:~/Downloads/avs/avs/tcp$ wine64 ./client_tcp.exe
Enter Message : hello
Message from server: hello

1 - Exit 2 - Continue :

```

Рисунок 2.2 – Запуск сервера и отправка сообщения от клиента на TCP

Приложение А

(обязательное)

Листинг модуля «Клиент TCP»

```
#include <stdio.h>
#include <stdlib.h>
#include <winsock2.h>
#include <ws2tcpip.h>

#define BUF_SIZE 1024

int main(void) {

    WSADATA ws;
    WSAStartup(MAKEWORD(2, 2), &ws);

    SOCKET sock;
    sock = socket(AF_INET, SOCK_STREAM, 0);

    SOCKADDR_IN addr_serv;
    memset(&addr_serv, 0, sizeof(addr_serv));

    addr_serv.sin_family = AF_INET;
    addr_serv.sin_port = htons(1234);
    inet_pton(AF_INET, "127.0.0.1", &addr_serv.sin_addr);

    connect(sock, (SOCKADDR*) &addr_serv, sizeof(addr_serv));

    char message[BUF_SIZE];
    char exit;
```

```
do {  
    printf("Enter Message : ");  
    fgets(message, BUF_SIZE, stdin);  
    send(sock, message, BUF_SIZE, 0);  
    recv(sock, message, BUF_SIZE, 0);  
    printf("Message from server: %s\n", message);  
    printf("1 - Exit 2 - Continue :");  
    scanf("%c", &exit);  
    char c;  
    do {  
        c = getchar();  
    } while (c != EOF && c != '\n');  
  
} while (exit != '1');  
  
closesocket(sock);  
  
return 0;  
}
```

Приложение Б

(обязательное)

Листинг модуля «Сервер TCP»

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

#define BUF_SIZE 1024
#define QUEUE_SIZE 50
#define MAX_COUNT 1000

void error(const char *msg) {
    perror(msg);
    exit(1);
}

void *start_func(void *arg) {
    printf("%s\n", "The connection is created!");
    char buffer[BUF_SIZE];
    while (1) {
        if (recv(((int*)arg), buffer, BUF_SIZE, 0) <= 0) {
            printf("%s\n", "The connection is disabled!");
            pthread_exit(NULL);
        }
    }
}
```

```

        send(*((int*)arg), buffer, BUF_SIZE, 0);
    }
}

int main(int argc, char *argv[]) {

    int sock_serv;
    socklen_t client_len;
    int port = 1234;
    if (argc == 2) {
        port = atoi(argv[1]);
    }

    struct sockaddr_in serv_addr;
    struct sockaddr_in client_addr;
    int n;
    sock_serv = socket(AF_INET, SOCK_STREAM, 0);
    if (sock_serv < 0) {
        error("ERROR opening socket");
    }
    memset(&serv_addr, 0, sizeof(serv_addr));

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(port);

    if (bind(sock_serv, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) <
0) {

        error("ERROR on binding");
    }
}

```



```

listen(sock_serv, QUEUE_SIZE);
client_len = sizeof(client_addr);

printf("%s\n", "The server works, waits for connecting!" );
pthread_t pthread_arr[MAX_COUNT];
int tmp_sock_client[MAX_COUNT];
int i = 0;
while(1) {
    if (i == MAX_COUNT) {
        break;
    }
    (tmp_sock_client[i]) = accept(sock_serv, NULL, NULL);
    pthread_create(&(pthread_arr[i]), NULL, start_func, (void *)
&(tmp_sock_client[i]));
    i++;
}

for (int j = 0; j < i; j++) {
    close(tmp_sock_client[j]);
    pthread_join(pthread_arr[j], NULL);
}

close(sock_serv);
return 0
}

```

Приложение В

(обязательное)

Листинг модуля «Сервер UDP»

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

#define BUF_SIZE 1024

void error(const char *msg) {
    perror(msg);
    exit(1);
}

int main(int argc, char *argv[]) {
    int sock_serv;
    socklen_t client_len;
    char buffer[BUF_SIZE];
    int port = 1234;
    if (argc == 2) {
        port = atoi(argv[1]);
    }

    struct sockaddr_in serv_addr;
```

```

struct sockaddr_in client_addr;
sock_serv = socket(AF_INET, SOCK_DGRAM, 0);
if (sock_serv < 0) {
    error("ERROR opening socket");
}
memset(&serv_addr, 0, sizeof(serv_addr));

serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = INADDR_ANY;
serv_addr.sin_port = htons(port);

if (bind(sock_serv, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0) {
    error("ERROR on binding");
}

client_len = sizeof(client_addr);

printf("%s\n", "The server works, waits for connecting!");
int n;
while (1) {
    n = recvfrom(sock_serv, buffer, BUF_SIZE, 0, (struct sockaddr
*)&client_addr, &client_len);
    if (n < 0) error("recvfrom");
    n = sendto(sock_serv, buffer, BUF_SIZE, 0, (struct sockaddr
*)&client_addr, client_len);
    if (n < 0) error("sendto");
    printf("%s\n", "Message sent");
}
return 0;
}

```

Приложение Г

(обязательное)

Листинг модуля «Клиент UDP»

```
#include <stdio.h>
#include <stdlib.h>
#include <winsock2.h>
#include <ws2tcpip.h>

#define BUF_SIZE 1024

int main(void)
{
    WSADATA ws;
    WSAStartup(MAKEWORD(2, 2), &ws);

    SOCKET sock;
    sock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
    struct hostent *hp;

    SOCKADDR_IN addr_serv, my_addr, from;
    memset(&addr_serv, 0, sizeof(addr_serv));
    int from_len, length = sizeof(addr_serv);

    addr_serv.sin_family = AF_INET;
    addr_serv.sin_port = htons(1234);
    inet_pton(AF_INET, "127.0.0.1", &addr_serv.sin_addr);

    my_addr.sin_family = AF_INET;
    my_addr.sin_port = htons(1200);
    inet_pton(AF_INET, "127.0.0.1", &my_addr.sin_addr);
```

```

if (bind(sock, (SOCKADDR*)&my_addr, sizeof(my_addr)) == 0)
    printf("Binded Correctly\n");

else
    printf("Unable to bind\n");

char message[BUF_SIZE];
char exit;
do {
    printf("Enter Message : ");
    fgets(message, BUF_SIZE, stdin);
    sendto(sock, message, BUF_SIZE, 0, (SOCKADDR*)&addr_serv, length);
    recvfrom(sock, message, BUF_SIZE, 0, (SOCKADDR*)&from, &from_len);
    printf("Message from server: %s\n", message);
    printf("1 - Exit 2 - Continue :");
    scanf("%c", &exit);
    char c;
    do {
        c = getchar();
    } while (c != EOF && c != '\n');

} while (exit != '1');

closesocket(sock);

return 0;
}

```

3 Заключение

В ходе лабораторной работы были изучены средства синхронизации потоков и процессов.

Исходные коды программ и Dockerfile приложены в архиве к отчету, а также загружены на GitHub (<https://github.com/7371avs/SP>).