Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Факультет Безопасности (ФБ)

Кафедра комплексной информационной безопасности электронно-вычислительных систем (КИБЭВС)

ПРОГРАММИРОВАНИЕ СОКЕТОВ

Отчёт по практической работе № 7 по дисциплине  
«Системное программирование»

Студент группы 726

\_\_\_\_\_\_\_\_\_\_ М.И. Васильева

\_\_\_\_\_\_\_\_\_\_

Принял

к.т.н., доцент кафедры КИБЭВС

\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ Я.Е. Мещеряков

оценка

\_\_\_\_\_\_\_\_\_\_

дата

1 Введение

Цель работы: Познакомиться с основными функциями POSIX API для работы с сокетами, особенностями сокетов в операционных системах Unix.

Ход работы:

1. Ознакомление с теоретическим материалом по работе с сокетами

2. Разработка и написание программ на языках Python3 и С++

3. Отладка программ

4. Оформление отчёта

5. Сдача лабораторной работы

Задание:

1. Изучить теоретические сведения, самостоятельно изучить способы

работы с сокетами в POSIX.

1. Разработать и написать программы на языке Python3 и С++, которые используют сокеты для решения задачи.
2. На moodle загрузить архив, содержащий: Отчет, все файлы проекта.
3. Загрузить все необходимые файлы в репозиторий на github.
4. Написать отчет и защитить у преподавателя.

2 Ход работы

В ходе работы были написаны 2 программы на языке Python3 и 2 программы на языке С++.

Результат работы серверной части программы на Python3, реализующей передачу данных с помощью сокета представлен на рисунке 1. Так как работа производится в Докер контейнере, то сначала сервер запускается в фоновом режиме и ждет подключения клиента, чтобы передать адрес соединение и сообщение от сервера.



Рисунок 1 – Результат работы клиентской программы на Python3

Результат работы клиентской части программы на Python3, реализующей передачу данных с помощью сокета представлен на рисунке 2. На рисунке видно сообщение от сервера об адресе соединения и переданное сообщение клиенту.

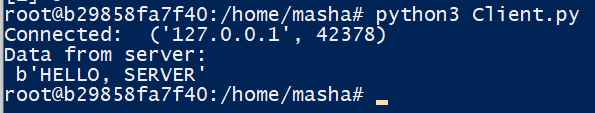


Рисунок 2 - Результат работы серверной программы на Python3

Результат работы серверной части программы на C++, реализующей передачу данных с помощью сокета представлен на рисунке 3. Так же как и при запуске серверной части, написанной на языке программирования Python, сервер запускается в фоновом режиме.



Рисунок 3 – Результат работы серверной программы на C++

Результат работы клиентской части программы на C++, реализующей передачу данных с помощью сокета представлен на рисунке 4. С клиентской части передается написанное пользователем сообщение. Сервер, работающий в фоновом режиме, получает данное сообщение и сообщает о том, что сообщение было получено.

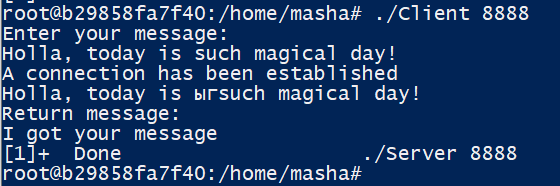


Рисунок 4 – Результат работы клиентской программы на C++

3 Заключение

В ходе работы были изучены с основные функции POSIX API для работы с сокетами на языке Python3 и С++, особенностями сокетов в операционных системах Unix.

Все необходимые файлы для воспроизведения работы данной практической работы были загружены в репозиторий: https://github.com/mariaigorevna/Socket.

Приложение А

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

**import** socket

sock **=** socket**.**socket**()**

sock**.**connect**((**'localhost'**,** 8000**))**

sock**.**send**(**"Hello, SERVER"**.**encode**())**

data **=** sock**.**recv**(**1024**)**

**print(**'Data from server:\n'**,** data**)**

sock**.**close

Листинг А.1 – Исходный код клиентской части Python программы

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

**import** socket

sock **=** socket**.**socket**()**

sock**.**bind**((**''**,** 8000**))**

sock**.**listen**(**1**)**

conn**,** addr **=** sock**.**accept**()**

**print** **(**'Connected: '**,** addr**)**

**while** **True:**

data **=** conn**.**recv**(**1024**)**

**if** **not** data**:**

**break**

conn**.**send**(**data**.**upper**())**

conn**.**close**()**

Листинг А.1 – Исходный код серверной части Python программы (Конец)

#include <sys/types.h>

#include <sys/socket.h>

#include <unistd.h>

#include <cstdlib>

#include <sys/un.h>

#include <iostream>

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

**{**

int sockfd**,** servlen**,** n**;**

struct sockaddr\_un serv\_addr**;**

char buffer**[**82**];**

bzero**((**char**\*)&**serv\_addr**,** **sizeof(**serv\_addr**));**

serv\_addr**.**sun\_family **=** AF\_UNIX**;**

strcpy**(**serv\_addr**.**sun\_path**,** argv**[**1**]);**

Листинг А.1 – Исходный код клиентской части C++ программы (Начало)

servlen **=** strlen**(**serv\_addr**.**sun\_path**)** **+** **sizeof(**serv\_addr**.**sun\_family**);**

**if** **((**sockfd **=** socket**(**AF\_UNIX**,** SOCK\_STREAM**,** 0**))** **<** 0**)**

**{**

std**::**cerr **<<** "Error creating socket\n"**;**

exit**(**EXIT\_FAILURE**);**

}

if (connect(sockfd, (struct sockaddr \*)&serv\_addr, servlen) < 0)

{

std::cerr << "Error connecting to server";

exit(EXIT\_FAILURE);

}

std::cout << "Enter your message:\n";

bzero(buffer, 82);

fgets(buffer, 80, stdin);

write(sockfd, buffer, strlen(buffer));

n = read(sockfd, buffer, 80);

std::cout << "Return message:\n";

write(1, buffer, n);

close(sockfd);

return 0;

}

Листинг А.1 – Исходный код клиентской части C++ программы (Конец)

#include <sys/types.h>

#include <sys/socket.h>

#include <unistd.h>

#include <cstdlib>

#include <sys/un.h>

#include <iostream>

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

**{**

int sockfd**,** newsockfd**,** servlen**,**n**;**

socklen\_t clilen**;**

struct sockaddr\_un cli\_addr**,** serv\_addr**;**

char buf**[**80**];**

**if** **((**sockfd **=** socket**(**AF\_UNIX**,** SOCK\_STREAM**,** 0**))** **<** 0**)**

**{**

std**::**cerr **<<** "Error creating socket\n"**;**

exit**(**EXIT\_FAILURE**);**

**}**

bzero**((**char**\*)** **&**serv\_addr**,** **sizeof(**serv\_addr**));**

serv\_addr**.**sun\_family **=** AF\_UNIX**;**

strcpy(serv\_addr.sun\_path, argv[1]);

servlen = strlen(serv\_addr.sun\_path) + sizeof(serv\_addr.sun\_family);

if (bind(sockfd, (struct sockaddr \*)&serv\_addr, servlen) < 0)

{

std::cerr << "Error binding socket\n";

exit(EXIT\_FAILURE);

}

listen(sockfd, 5);

clilen = sizeof(cli\_addr);

newsockfd = accept(sockfd, (struct sockaddr \*)& cli\_addr, &clilen);

if (newsockfd < 0)

{

Листинг А.1 – Исходный код серверной части C++ программы (Начало)

std::cerr << "Error accepting socket\n";

exit(EXIT\_FAILURE);

}

n = read(newsockfd, buf, 80);

std::cout << "A connection has been established\n";

write(1, buf, n);

write(newsockfd, "I got your message\n", 19);

close(newsockfd);

close(sockfd);

return 0;

}

Листинг А.1 – Исходный код серверной части C++ программы (Конец)